МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ОДЕСЬКА ПОЛІТЕХНІКА»
ІНСТИТУТ ЦИФРОВИХ ТЕХНОЛОГІЙ, ДИЗАЙНУ ТА ТРАНСПОРТУ
КАФЕДРА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ПРОЄКТУВАННЯ ТА ДИЗАЙНУ

КОНСПЕКТ ЛЕКЦІЙ

з дисципліни «Технології захисту інформації»
для студентів усіх форм навчання
(частина 2)

Перший (бакалаврський) рівень вищої освіти
Спеціальність – 122 КОМП'ЮТЕРНІ НАУКИ
Освітня програма – КОМП'ЮТЕРНИЙ ДИЗАЙН

Одеса – 2024

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ОДЕСЬКА ПОЛІТЕХНІКА»
ІНСТИТУТ ЦИФРОВИХ ТЕХНОЛОГІЙ, ДИЗАЙНУ ТА ТРАНСПОРТУ
КАФЕДРА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ПРОЄКТУВАННЯ ТА ДИЗАЙНУ

КОНСПЕКТ ЛЕКЦІЙ

з дисципліни «Технології захисту інформації»
для студентів усіх форм навчання
(частина 2)

Перший (бакалаврський) рівень вищої освіти
Спеціальність – 122 КОМП'ЮТЕРНІ НАУКИ
Освітня програма – КОМП'ЮТЕРНИЙ ДИЗАЙН

Затверджено на засіданні
кафедри інформаційних технологій
проєктування та дизайну
Протокол № 3 від 29.10.2024

Одеса – 2024

Конспект лекцій з дисципліни «Технології захисту інформації» для студентів першого (бакалаврського) рівня за спеціальністю 122 Комп’ютерні науки, за освітньо-професійною програмою «Комп’ютерний дизайн» (частина 2) / Укл.: А. С. Коляда, О. С. Лопаков, В. В. Космачевський – Одеса : Національний університет «Одеська політехніка», 2024. – 63 с.

Укладачі:

Коляда А. С., к.т.н., доц. кафедри
Лопаков О. С., ст. викладач
Космачевський В. В., ст. викладач

Даний конспект лекцій розглядає криптографічний захист інформаційних систем від викривлення та несанкціонованого використання даних. Інформація є цінним ресурсом, тому існують ризики зловмисних дій щодо систем, які її обробляють і передають. Проблема захисту інформації має давню історію, починаючи з військової та дипломатичної сфери, і сьогодні актуальна для бізнесу та приватних осіб. Розвиток комп’ютерних технологій суттєво змінив методи захисту. Криптографія стала важливою складовою мережевих технологій, забезпечуючи конфіденційність даних та контроль їхньої справжності. Зростання безпаперового документообігу, електронних платежів та цифрових архівів вимагає надійних механізмів захисту, адже відсутність фізичних підписів і печаток створює нові загрози. Водночас розвиток комп’ютерних технологій сприяє появі нових методів атак. Таким чином, для ефективного використання інформаційних систем необхідно розробляти та застосовувати сучасні засоби забезпечення конфіденційності та цілісності даних.

ЗМІСТ

ВСТУП

Даний курс лекцій присвячений проблемам криптографічного захисту інформаційних систем (ІС) від навмисних дій по викривленню та несанкціонованому використанню інформації, яка зберігається в них. Оскільки інформація може являти собою певну цінність, можливі різноманітні зловмисні дії по відношенню до систем, що зберігають, обробляють або передають таку інформацію.

Проблема захисту інформації має давню історію. Вона виникла з потреб таємної передачі, спочатку військових і дипломатичних повідомлень. В даний час вона актуальна в багатьох областях діяльності, в тому числі і для комерційних організацій і приватних осіб. Особливу актуальність проблема захисту інформації набула в інформаційних системах. Широке застосування комп'ютерів і комп'ютерних комунікацій радикально змінило характер і діапазон проблем захисту інформації.

Криптографія є в даний час невід'ємною частиною мережевих технологій. При використанні комп'ютерних мереж, по яким передаються великі обсяги інформації державного, комерційного, військового і приватного характеру, необхідно не допустити можливість доступу до цієї інформації сторонніх осіб.

Крім того, необхідно забезпечити контроль справжності інформації, що зберігається в електронному вигляді. Широке впровадження безпаперового документообігу також вимагає додаткових засобів захисту в силу відсутності на електронних документах підписів і печаток. Постає також питання про захист права на приватне життя, якщо в цьому житті використовуються електронна пошта, електронне зберігання особистих архівів. Багато хто користується такими криптографічними засобами, як шифрування електронної пошти, банківські картки та інше. Дедалі більшого поширення набувають безготівкові розрахунки з використанням телекомунікаційних мереж, електронні гроші. У той же час поява нових потужних комп'ютерів, технологій мережевих і нейронних обчислень зробило можливим дискредитацію серйозних систем захисту.

Таким чином, як при розробці ІС, так і при роботі з ними досить важливо вміти створювати і застосовувати ефективні засоби для реалізації всіх необхідних функцій, пов'язаних із забезпеченням конфіденційності та цілісності інформації.

Лекція №5 АСИМЕТРИЧНІ КРИПТОСИСТЕМИ

5.1 Причини виникнення асиметричних криптосистем

Появі нового напряму в криптології - асиметричної криптографії з відкритим ключем - сприяли дві проблеми, які не вдавалося вирішити в рамках класичної симетричної одноключової криптографії.

Перша з цих проблем пов'язана з *поширенням секретних ключів*. Як передати учасникам обміну інформацією змінювані секретні ключі, які потрібні їм для здійснення цього обміну? У загальному випадку для передачі ключа знову ж потрібне використання якоїсь криптосистеми, тобто завдання в рамках симетричної криптографії нерозв'язна.

Друга з цих проблем пов'язана з поширенням електронного документообігу. Виникла проблема забезпечення достовірності і авторства електронних документів. У звичайному, паперовому документообігу ця проблема вирішується за допомогою підпису на папері. Підробити підпис людини на папері зовсім не просто, а скопіювати ланцюжок цифр на ЕВМ - нескладна операція. Виникла проблема цифрового підпису, яка б виконувала всі ті завдання, які виконує підпис, поставлена на документі рукою. Обидві ці проблеми були успішно вирішені за допомогою криптографії з відкритими ключами. В опублікованій в 1976 р статтею "Нові напрямки в криптографії" У.Діффі і М.Хеллман вперше показали, що секретний зв'язок можливий без передачі секретного ключа між відправником і отримувачем.

На основі результатів, отриманих класичною та сучасною алгеброю, були запропоновані *системи з відкритим ключем*, називаються також *асиметричними криптосистемами*.

Суть їх полягає в тому, що кожним адресатом ІС генеруються два ключі, пов'язані між собою за певним правилом. Один ключ оголошується *відкритим*, а інший *закритим*. Відкритий ключ публікується і доступний кожному, хто бажає послати повідомлення адресату. Секретний ключ зберігається в таємниці. Якщо генератор ключів розташувати на стороні одержувача, то відпадає необхідність пересилання секретного ключа по каналу зв'язку.

Вихідний текст шифрується відкритим ключем адресата і передається йому. Зашифрований текст *в принципі не може бути розшифрованим* тим же відкритим ключем. Дешифрування повідомлення можливо тільки з використанням закритого ключа, який відомий тільки самому адресату. Таким чином, не потрібен секретний канал зв'язку для передачі ключа, але необхідно забезпечити справжність відкритого ключа, так як його викривлення або підміна не дозволить розшифрувати інформацію на парному з ним закритому ключі. Крім того, заміна зловмисником законного відкритого ключа на свій відкритий ключ надає йому повний доступ до шифруємої інформації.

5.2 Узагальнена схема асиметричної криптосистеми

Нижче (рис.5.1) наведена *узагальнена схема асиметричної криптосистеми*.

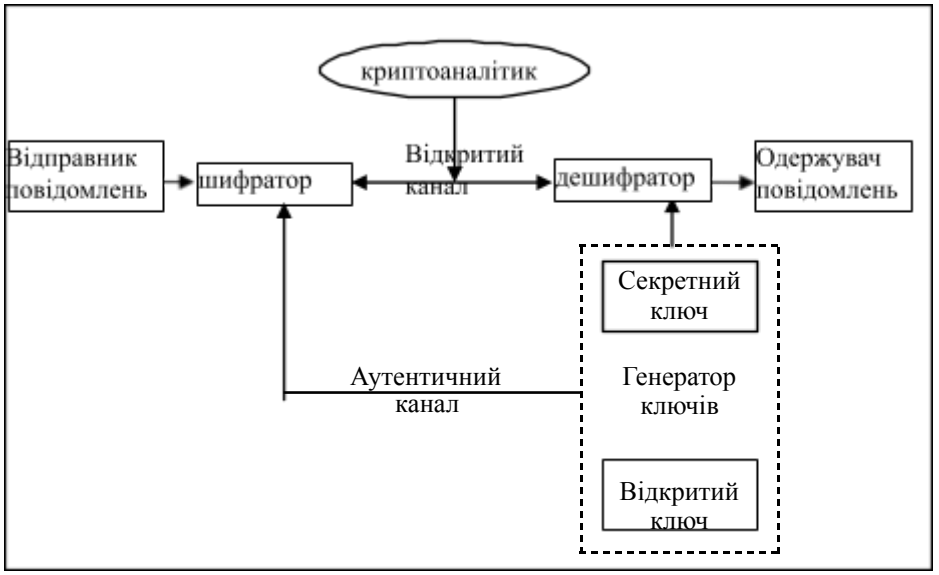


Рис. 5.1. Узагальнена схема асиметричної криптосистеми

Тут для передачі ключа використовується відкритий канал зв'язку, що забезпечує автентичність інформації, що передається.

Щоб гарантувати надійний захист інформації, до систем з відкритим ключем (СВК) пред'являються дві важливі і очевидні вимоги:

1 Перетворення вихідного тексту повинно бути необоротним і виключати його відновлення на основі відкритого ключа.

2 Визначення закритого ключа на основі відкритого також повинно бути неможливим на сучасному технологічному рівні. При цьому бажана точна нижня оцінка складності розкриття шифру.

Алгоритми шифрування з відкритим ключем набули широкого поширення в сучасних інформаційних системах. Їх використовують в наступних основних напрямках:

1. Як самостійні засоби захисту переданих і збережених даних.

2. Як засоби для розподілу криптографічних ключів. Алгоритми асиметричних криптосистем більш трудомісткі, ніж традиційні криптосистеми. Тому часто на практиці має сенс за допомогою асиметричних криптосистем розподіляти ключі, обсяг яких незначний. А потім за допомогою менш трудомістких симетричних алгоритмів здійснювати обмін великими інформаційними потоками.

3. Як засоби аутентифікації користувачів інформаційних систем, в тому числі для вирішення проблеми електронного підпису.

4. Як засоби для побудови складних криптографічних протоколів для вирішення різних завдань захисту інформації в сучасних інформаційних системах.

5.3 Алгебраїчна узагальнена модель шифру

Раніше ми розглядали алгебраїчну модель шифру К.Шеннона як трьохосновну універсальну алгебру A = (M, K, C, E), де

M - множина відкритих текстів,

K - множина ключів,

C - множина криптограми і

E - ін'єктивна (взаємно однозначна) функція шифрування:

E_k: MxK → C $E_k(m) = c$, де $m \in M$, $k \in K$, $c \in C$

Для того, щоб ця модель могла бути застосовна до асиметричних криптографічних систем, необхідно її розширення. Основна концептуальна ідея побудови такої моделі, природна і очевидна. Вона складається в окремому описі моделей двох шифрів: шифру шифрування і шифру розшифрування, сукупність яких і складає узагальнену алгебраїчну модель шифру.

Шифром шифрування (алгеброю зашифрування) назвемо алгебру:

$A_ш = (M, M_c, K_ш C, C_c, E)$, де

- множина $M_c \subseteq M$ трактується як підмножина всіх змістовних текстів з множини «відкритих текстів» M.
- функція шифрування E здійснює відображення $MxK_ш$ на C:
  $E: MxK_ш \rightarrow C$, $E_к(m) = c$

тобто є сюр'єктивною, причому $\forall k \in K_ш$ відображення $E_k(m)$ ін'єктивне (образи двох різних елементів різні), а множина C_c складається з шифрограм, які можуть бути отримані в результаті шифрування змістовних текстів: $C_c = E(M_c x K_ш)$, тобто результатів шифрування тих відкритих текстів m, для яких визначено значення $E_k(m)$ для всіх ключів шифру $k \in K_ш$.

Введення підмножини $M_c \subseteq M$ як множини змістовних текстів дозволяє коректно вводити критерії на змістовні тексти.

Таким чином, шифр шифрування є деяке уточнення моделі шифру Шеннона A = (M, $K_ш$, C, E).

Шифром розшифрування (алгеброю розшифрування) для $A_ш$ назвемо алгебру:

$A_p = (M_p, K_p, C_p, D)$, де

- $C \subseteq C_p$ і введення множини C_p - шифртекст «правильних» повідомлень (а точніше, $C_p \backslash C$ - множини перекручених шифртекстів) забезпечує можливість опису реакції приймальної сторони на вступ викривленого шифрованого повідомлення $c_p \notin C$;

- $M \subseteq M_p$, і введення множини $M_P \backslash M$ забезпечує можливість опису результату розшифрування приймальною стороною викривленого шифрованого повідомлення $c_p \notin C$;

- функція дешифрування D - сюр'єктивне відображення: D: $C_p x K_p \rightarrow M_p$, $D_k(c) = m$,

для якого виконуються наступні умови:

- існує бієкція f: $K_ш \rightarrow K_p$;

- для будь-яких $m \in M$, $k \in K_ш$ з умови $E_k(m) = c$ випливає: $D(c, f(k)) = m$

При відсутності викривлень в каналі зв'язку функція розшифрування D повністю визначена на всій множині $C \times K_p$.

Відзначимо, що в визначенні шифру розшифрування не міститься вимог ін'єктивності функції f по змінній $k \in K_p$.

Алгебраїчною узагальненою моделлю шифру назвемо трійку:
$(A_ш, A_p, f)$

До позитивних властивостей цієї моделі належить можливість моделювання шифрів як з симетричним, так і з асиметричним ключем.

При цьому враховуються такі міркування:

- ключ $k_ш \in K_ш$ не є таємним, а ключ $k_p = f(k_ш) \in K_p$ є секретним;
- визначення значення k пов'язано з вирішенням складних проблем;
- синтез пар ключів $(k_ш, k_p)$ проводиться досить просто.

Зауважимо, що тут проявляється можливість класифікації шифрів по параметру складності обчислення значення $f(k_ш)$ ключа розшифрування, що визначає основний параметр криптографічного стійкості шифрів з асиметричним ключем.

5.4 Односторонні функції

Концепція асиметричних криптографічних систем з відкритим ключем засновано на застосуванні односпрямованих або односторонніх функцій. Остання назва була дана по асоціації з одностороннім рухом, коли легко проїхати в одну сторону і не можна в іншу. При цьому в криптографії, як в житті, «не можна» не означає «неможливо ні за яких умов», але говорить про те, що це пов'язане з серйозними труднощами.

Визначення. Функція $f: X \to Y$ називається односторонньою (*oneway function*), якщо існує ефективний алгоритм для обчислення $f(x)$ $\forall x$, але не існує ефективного алгоритму для обчислення хоча б одного елемента прообразу $f^{-1}(y)$.

Ніхто не знає, чи існують взагалі односторонні функції. Основним критерієм віднесення функції f до класу односторонніх або незворотних є відсутність ефективних з *обчислювальної точки зору* алгоритмів зворотного перетворення $Y \to X$. У криптографії під *необоротністю* розуміється не теоретична необоротність функції, а практична неможливість обчислити зворотнє значення, використовуючи сучасні обчислювальні засоби за заданий інтервал часу. Таким чином, проблеми побудови односторонніх функцій пов'язані з теоретико-ймовірнісною складністю алгоритмів і алгоритмічними питаннями теорії чисел.

Безліч класів необоротних функцій і породжує все розмаїття систем з відкритим ключем. Більшість пропонованих сьогодні криптосистем з відкритим ключем спираються на один з наступних типів необоротних перетворень:

2 Розпад великих цілих чисел на прості множники.

3 Обчислення логарифма в кінцевому полі.

4 Обчислення коренів алгебраїчних рівнянь.

5.4.1 Факторизація

Як перший приклад односпрямованої функції розглянемо цілочислове множення. Пряма задача - обчислення добутку двох дуже великих цілих чисел p і q, тобто знаходження значення $n = p \cdot q$, є відносно нескладним завданням.

Зворотнє завдання, звана завданням факторизації, - розкладання на множники великого цілого числа, тобто знаходження дільників p і q великого цілого числа

$n = p/q$

є практично нерозв'язним завданням при досить великих значеннях n. За сучасними оцінками теорії чисел при цілому $n \approx 2^{664}$ і $p \approx q$ для розкладання числа n буде потрібно близько 10^{23} операцій, тобто задача практично нерозв'язна на сучасних ЕВМ.

Якщо прості множники мають спеціальний вид, відомі більш ефективні алгоритми факторизації. Йдеться про співмножників p, таких, у яких величини p-1 або p+1 є «гладкими», тобто мають тільки малі прості дільники.

Однак з появою алгоритму факторизації з використанням еліптичних кривих клас чисел, що допускають швидку факторизацію, розширився і прості критерії перевірки приналежності даного класу втратили свою важливість. Тому, як правило, єдиним розумним критерієм може слугувати розмір простих множників, оскільки зі збільшенням розміру зменшується ймовірність вибрати число спеціального виду.

5.4.2 Дискретний логарифм

Інший приклад односпрямованої функції - це модульна експонента з фіксованою підставою і модулем. Нехай a і n - цілі числа, такі, що $1 \le a \le n$. Тоді модульна експонента з підставою a за модулем n являє собою функцію:

$y = a^x \bmod n$

де x - ціле число. Звичайно записати $x = \log_a (y)$.

Завдання звернення цієї функції в множині цілих чисел називають завданням знаходження дискретного логарифма.

Визначення. Число x називають дискретним логарифмом числа y за основою a і модулю n, якщо для всіх $a \in Z_n$ знайдеться таке ціле y, що

$y = a^x \bmod n$

Обчислення дискретних логарифмів (коли задані a, y і n) приблизно така ж важко вирішуване завдання, як і розкладання на множники.

Визначення. Одностороння функція f: X → Y називається *односторонньою функцією з пасткою*, якщо $f^{-1}(y)$ можна обчислити за поліноміальний час, маючи деяку додаткову інформацію, тобто існує функція $g(y, t)$, обчислювана за поліноміальний час і така , що $g(y, t) = f^{-1}(y)$ для деякої пастки t.

Ефективне обчислення оберненої функції можливо, якщо відомий "потайний хід" (секретне число, рядок або інша інформація, що асоціюється з цією функцією). Як приклад односпрямованої функції з "потайним ходом" можна привести використання функції Ейлера в криптосистемі RSA.

5.5 Криптосистема RSA

Алгоритм RSA став першим повноцінним алгоритмом з відкритим ключем, який може працювати як в режимі шифрування даних, так і в режимі електронного цифрового підпису. Заснована на цьому алгоритмі популярна криптосистема RSA розроблена в 1977 році і отримала назву на честь її творців: Рональда Рівеста (в даний час він очолює компанію RSA Data Security), Аді Шаміра і Леонарда Ейдельмана.

В даний час RSA є найбільш поширеною криптосистемою з відкритим ключем - стандартом де-факто для багатьох криптографічних додатків. Статус де-факто став причиною включення криптосистеми RSA в прийняті раніше криптографічні стандарти, наприклад в фінансові стандарти США і Франції, австралійський стандарт управління ключами і багато інших. Криптосистема RSA застосовується в різних протоколах Internet. У криптографічні стандарти, що діють на території СНГ, RSA не входить, що ускладнює її застосування з точки зору правових норм.

5.5.1 Основні визначення і теореми

Надійність алгоритму ґрунтується на важко обчислюваних завданнях факторизації (розкладання на множники) великих чисел і обчислення дискретних логарифмів. Визначення та теореми з алгебри, використані при створенні даної криптосистеми розглянуті в додатку. Наведемо тут лише основні твердження.

- Криптографічні системи є стійкими, якщо певні їх параметри є ***простими числами***. Число *a* називається простим, якщо воно не має цілих дільників, крім одиниці. Числа *a* і *b* називаються взаємно простими, якщо їх найбільший спільний дільник НСД (*a*, *b*) = 1.
- ***Функцією Ейлера*** φ (*n*) називається число позитивних цілих чисел менших *n* і взаємно простих з *n*.

Обчислення функції Ейлера φ (*n*) для великих *n* в загальному випадку являє собою трудомістку процедуру перебору всіх чисел менших *n* і перевірки для кожного взаємної простоти з *n*. Однак, ця функція має такі властивості:

1 φ (*p*) = *p* - 1 ∀*p* - простого числа.

2 φ (*a*, *b*) = φ (*a*) φ (*b*) для будь-яких натуральних взаємно простих *a* й *b*,

які дозволяють легко обчислити значення функції Ейлера φ (*n*) за допомогою трьох арифметичних дій, якщо відомо розкладання числа *n* на прості множники *p* і *q*:

φ (*n*) = (*p*-1) (*q*-1).

- У RSA використовується теорема, яка носить назву ***китайської теореми про залишки***, так як цей результат був відомий ще в древньому Китаї, де теорема була запропонована китайським математиком першого століття Сун Це. Вона стверджує, що будь-яке невід'ємне ціле число, яке не перевищує добутку модулів, можна однозначно відновити, якщо відомі його відрахування по цих модулям, і названа так тому, що результатом приведення числа *a* по модулю *n* є залишок від ділення *a* на *n*. Фактично в RSA використовується наслідок з цієї теореми, яке стверджує, що якщо відомо розкладання числа *n* на прості множники $n=n_1n_2...n_k$, де все n_i попарно взаємно прості, і результат приведення числа *x* по модулю n_i $\forall i=1,\ldots,k$ однаковий, то результатом приведення числа *x* по модулю *n* буде те саме число. Тобто ∀*x*, *a* - цілих чисел:

$$x \equiv a \bmod n \;\Leftrightarrow\; x \equiv a \bmod n_i \;\forall i=1,\ldots,k$$

- ***Теорема Ейлера***. Якщо $n > 1$, то $\forall x \in Z_n^*$ (*x* взаємно простого з *n*), виконується порівняння

$$x^{\varphi(n)} \equiv 1 \bmod n$$

Слідство. $\forall x < n$ і взаємно простого з *n* можна легко обчислити зворотний елемент x^{-1} в кільці відрахувань Z_n з порівняння:

$$x^{-1} \equiv x^{\varphi(n)-1} \bmod n$$

Мала теорема Ферма. $\forall x \in GF(p),\ x \neq 0$, виконується порівняння:

$$x^{p-1} \equiv 1 \bmod p$$

Мала теорема Ферма є наслідком з теореми Ейлера, хоча історично вона була доведена раніше, потім Ейлер її узагальнив.

Слідство 1: Якщо p - просте число, то $\forall x$, взаємно простого з p: $x^p = x \bmod p$

Слідство 2: якщо НСД (e, φ (n)) = 1 (e - просте відносно φ (n))

то ∃d-ціле, таке, що:

$ed = 1 \bmod n$

На цих математичних фактах заснований алгоритм RSA.

5.5.2 Алгоритм RSA

У криптосистемі RSA повідомлення m, криптограма c, відкритий ключ K_o, і секретний ключ K_c, належать множині цілих чисел $Z_n = \{0, 1, 2, ..., n-1\}$. Безліч Z_n з операціями додавання і множення по модулю n утворює кільце.

Модуль n визначається як складене число, яке дорівнює добутку $n = p \cdot q$ двох великих простих чисел p і q. Модуль n є відкритим параметром алгоритму, а числа p і q - секретними параметрами. Тобто множники p і q зберігають в секреті, а їх добуток n відомо всім, хто користується цією криптосистемою. Тут використовується одностороння функція з пасткою. Знаючи секретні параметри алгоритму p і q можна легко обчислити функцію Ейлера $\phi(n)$ за формулою:

$\phi(n) = (p\text{-}1)(q\text{-}1)$

тоді як обчислення $\phi(n)$ тільки по великому числу n є важко обчислюваним завданням.

Функція Ейлера використовується в RSA при обчисленні ключів.

Відкритий ключ $K_o = e$ вибирають випадковим чином з множини:

$Z^*_{\phi(n)}$ - чисел менших $\phi(n)$ і взаємно простих з $\phi(n)$.

Інакше умова $e \in Z^*$ рівносильне виконанню двох умов:

$1 < K_o \le \phi(n)$, и НОД (K_o, $\phi(n)$) = 1

яким повинно задовольняти випадково вибране число $K_o = e$, щоб воно могло служити відкритим ключем в RSA.

Секретний ключ $K_c = d$ обчислюють так, щоб виконувалася умова:

$K_c \cdot K_o = e \cdot d \equiv 1 \bmod \phi(n)$ (*)

Тобто, секретний ключ d є зворотним елементом до відкритого ключа e в множині $Z_{\phi(n)}$: $d = e^{-1} \bmod \phi(n)$. Рішення порівняння (*) можна знайти за допомогою розширеного алгоритму Евкліда.

Зауважимо, що d і n також взаємно прості числа. Обчислення секретного ключа з відкритого є важко обчислюваним завданням, якщо невідомі секретні параметри алгоритму p і q, так як при цьому важко обчислити значення функції Ейлера, тобто модуля, за яким наводяться результати операцій при обчисленні ключів.

При виконанні шифрування і дешифрування обчислення наводяться по модулю n. Відкритий ключ K_o і модуль n повідомляють всім, з ким припускають обмінюватися повідомленнями і використовують для шифрування даних, а секретний ключ K_c зберігають в секреті на стороні одержувача і використовують для дешифрування.

Перетворення шифрування визначає криптограму c через пару (відкритий ключ K_o, повідомлення m) у відповідності з наступною формулою:

$c = E_{K_o}(m) = m^{K_o} \bmod n$

Як алгоритм швидкого обчислення значення c використовують ряд послідовних зведень в квадрат цілого m і множень на m з приведенням по модулю n.

Звернення функції $c = m^{K_o} \bmod n$, тобто визначення значення m за відомими значеннями c, K_o і n, є завданням дискретного логарифмування і практично нездійсненно при $n \approx 2^{512}$.

Однак завдання розшифрування криптограми c, можна легко вирішити, використовуючи секретний ключ K_c, за такою формулою:

$m = D_{K_c}(c) = c^{K_c} \bmod n$

Доведемо, що в результаті зведення криптограми c в ступінь секретного ключа K_c виходить вихідний текст m. Процес розшифрування можна записати так:

$D_{K_c}(E_{K_o}(m)) = D_{K_c}(m^e) \bmod n = (m^e)^d \bmod n = m^{ed} \bmod n$

За умовою вибору ключів:

$e \cdot d \equiv 1 \bmod \phi(n) \quad (*)$

ми можемо написати, що $\exists\, k$ таке що, з урахуванням властивостей $\phi(n)$:

$e \cdot d = k \cdot \phi(n) + 1 = k\,(p-1)(q-1) + 1$

Підставимо цей вираз в показник ступеня:

$m^{ed} \equiv m\,(m^{(p-1)})^{k\,(q-1)} = (m^{(q-1)})^{k\,(p-1)}$

За малої теореми Ферма ($x^{p-1} \equiv 1 \bmod p$, p - просте):

$m^{ed} \bmod p \equiv m\,(1)^{k\,(q-1)} \bmod p = m \bmod p$

Аналогічно, замінивши p на q, отримаємо:

$m^{ed} \bmod q \equiv m\,(1)^{k\,(p-1)} \bmod q = m \bmod q$

Далі, по слідству з китайської теореми про залишки так як $n = p \cdot q$:
$m^{ed} \bmod n = m \bmod n$, ч.т.д.

Таким чином, якщо криптограму з звести в ступінь K_c:

$c^{K_c} \bmod n = m$

то в результаті відновлюється вихідний відкритий текст m.

Саме тому для обчислення секретного ключа K_c використовують співвідношення (*).

5.5.3 Процедури шифрування і розшифрування в криптосистемі RSA

У реальних системах алгоритм RSA реалізується в такий спосіб. Припустимо, що користувач А хоче передати користувачеві В повідомлення в зашифрованому вигляді, використовуючи криптосистему RSA. У такому випадку користувач А виступає в ролі відправника повідомлення, а користувач В - в ролі одержувача. Криптосистему RSA повинен сформувати одержувач повідомлення, тобто користувач В, так як в цьому випадку не буде необхідності в передачі секретного ключа. Розглянемо послідовність дій користувача В і користувача А.

Формування криптосистеми (на стороні одержувача інформації) полягає у виборі параметрів алгоритму та обчисленні пари ключів:

- Користувач В вибирає два великих простих числа p і q. Це секретні параметри алгоритму, вони зберігаються в секреті на стороні одержувача.

- Користувач В обчислює значення модуля n криптосистеми як результат множення перших двох чисел:

  $n = p \cdot q$

  Це *загальнодоступний параметр* криптосистеми. Іноді його включають у відкритий ключ.

- Користувач В, знаючи секретні параметри алгоритму p і q, обчислює функцію Ейлера:

  $\phi(n) = (p-1)(q-1)$

  і вибирає випадковим чином просте число e як значення відкритого ключа $К_о$ з урахуванням виконання умов:

  $К_о < \phi(n)$ и НОД ($К_о, \phi(n)$) = 1

- Користувач В обчислює значення секретного ключа $К_с = d$ при вирішенні порівняння

  $К_с \equiv К_о^{-1} \bmod \phi(n)$

Для обчислення зворотного елемента в кільці $Z_{\phi(n)}$ використовують приватний режим роботи розширеного алгоритму Евкліда для визначення НОД. Це можна здійснити, так як одержувач В знає пару простих чисел (p, q) і може легко знайти $\phi(n)$. Зауважимо, що $К_с$ і n повинні бути взаємно простими.

(В принципі відкриті і закриті ключі можна поміняти місцями, головне, щоб виконувалося співвідношення $e \cdot d \equiv 1 \bmod \phi(n)$).

- Користувач В пересилає користувачеві А пару чисел $\{e, n\}$ по незахищеному каналу.

Шифрування інформації (на стороні відправника). Якщо користувач А хоче передати користувачеві В повідомлення m, він виконує наступні кроки.

- Користувач А розбиває вихідний відкритий текст m на блоки m_i, $i = 1, ..., N$, кожен з яких може бути представлений у вигляді числа меншого n

  $m_i \in \{0, 1, 2, ... , n-1\}$

- Користувач А шифрує текст, представлений у вигляді послідовності чисел m_i за допомогою ключа $К_о = e$ за формулою:

  $c_i = m_i^e \bmod n$

  і відправляє криптограму $c_1, ... , c_i$ користувачу В.

Дешифрування інформації (на стороні одержувача):

- Користувач В розшифровує прийняту криптограму $c_1, ... , c_i$, використовуючи секретний ключ $К_с = d$, за формулою

  $m_i = c_i^d \bmod n$

В результаті буде отримана послідовність чисел, які представляють собою вихідне повідомлення m.

Таким чином, коли одержувач В, який створює криптосистему, він захищає наступні параметри:

- секретний ключ Кс;
- пару чисел (p, q);

Відкритий ключ K_o і значення модуля n публікуються і доступні кожному, хто бажає послати власнику ключа повідомлення, яке зашифровується зазначеним алгоритмом. Після шифрування повідомлення неможливо розкрити за допомогою відкритого ключа. Власник закритого ключа без праці може розшифрувати прийняте повідомлення. Противнику ж для того, щоб визначити значення секретного ключа K_c потрібно зуміти розкласти число n на множники p і q, щоб дізнатися про "потайний хід" і обчислити значення функції Ейлера як $\phi(n) = (p-1)(q-1)$.

Приклад. Розглянемо невеликий приклад, який ілюструє застосування алгоритму RSA. Зашифруємо повідомлення "CAB". Для простоти будемо використовувати маленькі числа (на практиці застосовуються набагато більші).

- Виберемо $p = 3$ і $q = 11$.

- Визначимо $n = 3 \cdot 11 = 33$.

3. Знайдемо $\phi(n) = (p-1)(q-1) = 2 \cdot 10 = 20$.

- Виберемо в якості секретного ключа d довільне число взаємно просте з $\phi(n) = 20$,

наприклад, $d = 3$.

- Виберемо відкритий ключ $e = 7$. В якості такого числа може бути взято будь-яке число, для якого задовольняється співвідношення

$e \cdot d \mod \phi(n) = 7 \cdot 3 \mod 20 = 1$

- Уявімо шифруємо повідомлення як послідовність цілих чисел за допомогою відображення: A = 1, B = 2, C = 3. Тоді повідомлення приймає вид (3,1,2). Зашифруємо повідомлення за допомогою ключа $\{e = 7, n = 33\}$:

ШТ1 = $(3^7) \mod 33 = 2187 \mod 33 = 9$,

ШТ2 = $(1^7) \mod 33 = 1 \mod 33 = 1$,

ШТ3 = $(2^7) \mod 33 = 128 \mod 33 = 29$.

- Розшифруємо отримане зашифроване повідомлення (9, 1, 29) на основі закритого ключа $\{d = 3, n = 33\}$:

ІТ1 = $(9^3) \mod 33 = 729 \mod 33 = 3$,

ІТ2 = $(1^3) \mod 33 = 1 \mod 33 = 1$,

ІТ3 = $(29^3) \mod 33 = 24389 \mod 33 = 2$.

Надійність RSA. RSA багато років протистоїть інтенсивному криптоаналізу. Доведено, що розкриття шифру RSA еквівалентно рішенню завдання факторизації великих (100-200 двійкових розрядів) чисел. Важливо, що в цьому завданні для будь-якої довжини ключа можна дати нижню оцінку числа операцій для розкриття шифру, а з урахуванням продуктивності сучасних комп'ютерів оцінити і необхідний на це час.

Можливість гарантовано оцінити захищеність алгоритму RSA стала однією з причин популярності цієї СВК на фоні десятків інших схем. Тому алгоритм RSA використовується в банківських комп'ютерних мережах, особливо для роботи з віддаленими клієнтами (обслуговування кредитних карток). В даний час алгоритм RSA активно реалізується як у вигляді самостійних криптографічних продуктів, так і в якості вбудованих засобів в популярних додатках, а також використовується в багатьох стандартах.

5.6 Криптосистема Ель-Гамаля

Дана система є альтернативою RSA і при рівному значенні ключа забезпечує ту ж криптостійкість. Однак спільної думки з приводу переваги того чи іншого методу немає.

На відміну від RSA метод Ель-Гамаля заснований на проблемі дискретного логарифма. Цим він схожий на *алгоритм Діффі-Хелмана*. Якщо зводити число в ступінь в кінцевому полі досить легко, то відновити аргумент за значенням (тобто знайти логарифм в множині цілих чисел) досить важко.

Основу системи складають параметри p і g - числа, перше з яких p - просте, а друге ($g \in Z_p$) - ціле. Дані параметри не є секретними і можуть бути загальними для групи користувачів. У реальних схемах шифрування необхідно використовувати в якості модуля велике ціле просте число, що має в двійковому поданні довжину 512 ... 1024 біт.

Секретний ключ $x \in Z_{p-1}$ генерується випадковим чином, а відкритий ключ y обчислюється за формулою:

$$y = g^x \bmod p$$

Для шифрування повідомлення m спочатку вибирається випадкове число k, взаємно просте з p-1, яке називають також сеансовим ключем.

Шифртекст є пара чисел (a, b), що обчислюється за формулами:

$$a = g^k \bmod p$$

$$b = y^k m \bmod p$$

Таким чином, шифртекст в два рази довший відкритого тексту.

Для дешифрування обчислюється:

$$m = \frac{b}{a^x} \bmod p$$

Перетворення можна зупинити, так як:

$$a^x = g^{kx} \bmod p$$

$$\frac{b}{a^x} \equiv \frac{y^k m}{a^x} \equiv \frac{g^{xk} m}{a^x} = m \bmod p$$

Число k називають також рандомізатором. Його використання означає, що тут реалізований багатозначний шифр заміни. При цьому для шифрування різних блоків (чисел) відкритого тексту необхідно використовувати різні значення рандомізатора. При використанні одного і того ж значення відповідні шифртекст (a, b) і (a', b'), отримані для блоків відкритих текстів m і m', пов'язані співвідношенням:

$$b(b')^{-1} = m(m')^{-1}$$

і текст m' можна обчислити, якщо відомий текст m.

Стійкість криптосистеми Ель Гамаля заснована на складності завдання логарифмування в мультиплікативній групі кінцевого простого поля. Ця криптосистема може бути узагальнена для застосування в будь-якій кінцевій циклічної групи. В якості такої групи, крім розглянутої Z^*_p, найчастіше використовується мультиплікативна група кінцевого поля $GF(2m)$ і група точок на еліптичній кривій над кінцевим полем.

Алгоритм не запатентований, але потрапляє під дію патенту на метод експоненціального ключового обміну Діффі-Хеллмана, розглянутий нижче. Перетворення шифрування/дешифрування Ель Гамаля по суті те ж саме, що ключовий обмін по Діффі-Хеллману, за винятком того, що y - це частина ключа, а при шифруванні повідомлення множиться на y^k. Алгоритм цифрового підпису DSA, розроблений NIST (National Institute of Standard and Technology USA) і є частиною стандарту DSS частково спирається на розглянутий метод.

5.7 Комбінований метод шифрування

Головною перевагою криптосистем з відкритим ключем є їх потенційно висока безпека: немає необхідності ні передавати, ні повідомляти будь-кому значення секретних ключів, ні переконуватися в їх справжності. У симетричних криптосистемах існує небезпека розкриття секретного ключа під час передачі.

Однак алгоритми, що лежать в основі криптосистем з відкритим ключем, мають такі недоліки:

- генерація секретних і відкритих ключів заснована на генерації великих простих чисел, а перевірка простоти чисел займає багато процесорного часу;
- процедури шифрування і розшифрування, пов'язані зі зведенням до ступеня багатозначного числа, досить трудомісткі.

Тому швидкодію (швидкість шифрування і дешифрування) в криптосистемах з відкритим ключем зазвичай в сотні і тисячі разів менше швидкодії симетричних криптосистем з секретним ключем.

Комбінований метод шифрування дозволяє поєднувати переваги високої секретності, що надаються асиметричними криптосистемами з відкритим ключем, з перевагами високої швидкості роботи, властивими симетричним криптосистемам з секретним ключем. При такому підході асиметричні алгоритми шифрування застосовуються для шифрування, передачі і подальшого розшифрування тільки секретного ключа симетричної криптосистеми. А симетрична криптосистема застосовується для шифрування і передачі вихідного відкритого тексту. В результаті асиметричні алгоритми шифрування не замінює симетричну криптосистему з секретним ключем, а лише доповнює її, дозволяючи підвищити в цілому захищеність переданої інформації.

Користувачі А і В, які використовують комбінований метод шифрування, мають кожен по парі асиметричних ключів шифрування

($К_{Ao}$, $К_{Ac}$) и ($К_{Bo}$, $К_{Bc}$)

Якщо користувач А хоче передати зашифроване комбінованим методом повідомлення *m* користувачеві В, то порядок його дій буде такий.

- Створити (наприклад, згенерувати випадковим чином) симетричний ключ, званий в цьому методі сеансовим ключем K_s.

- Зашифрувати повідомлення *m* на сеансовому ключі K_s.
- Зашифрувати сеансовий ключ K_s на відкритому ключі $К_{Bo}$ користувача В і своєму таємному ключі $К_{Ac}$.

- Передати по відкритому каналу зв'язку на адресу користувача В зашифроване повідомлення разом із зашифрованим сеансовим ключем.

Вирішення В при отриманні зашифрованого повідомлення і зашифрованого сеансового ключа повинні бути зворотними:

- Розшифрувати на своєму таємному ключі $К_{Bc}$ і відкритий ключ $К_{Ao}$ користувача А сеансовий ключ K_s.

- За допомогою отриманого сеансового ключа K_s розшифрувати і прочитати повідомлення *m*.

При використанні комбінованого методу шифрування можна бути впевненим в тому, що тільки користувач В зможе правильно розшифрувати ключ K_s і прочитати повідомлення *m*.

Вибір довжини ключів в комбінованому методі шифрування. Таким чином, при комбінованому методі шифрування застосовуються криптографічні ключі як симетричних, так і асиметричних криптосистем. Очевидно, вибір довжин ключів для кожного типу криптосистеми слід здійснювати таким чином, щоб зловмисникові було

однаково важко атакувати будь-який механізм захисту комбінованої криптосистеми.

У таблиці 4.1. наведені поширені довжини ключів симетричних і асиметричних криптосистем, для яких труднощі атаки повного перебору приблизно дорівнює складності факторизації відповідних модулів асиметричних криптосистем.

Таблиця 4.1. Довжини ключів для симетричних і асиметричних криптосистем при однаковій їх криптостійкості

Симетричні криптосистеми	56	64	80	112	128
Асиметричні криптосистеми	384	512	786	1792	2304

5.8 Метод експоненціального ключового обміну Діффі-Хеллмана

Одні з авторів ідеї криптосистем з відкритим ключем, Діффі і Хеллмана запропонували нову ідею - відкритий розподіл ключів.

Вони задалися питанням: чи можна організувати таку процедуру взаємодії абонентів А і В по відкритих каналах зв'язку, щоб вирішити такі завдання:

- спочатку у А і В немає ніякої спільної секретної інформації, але в кінці процедури така загальна секретна інформація (загальний ключ) у А і В з'являється, т. Е. Виробляється;

- пасивний супротивник, який перехоплює все передачі інформації і знає, що хочуть отримати А і В, проте не може відновити вироблений загальний ключ А і В.

Метод отримав назву методу експоненціального ключового обміну. Він був першою криптосистемою з відкритим ключем, хоча для обміну ключами можна використовувати будь-які асиметричні алгоритми шифрування, наприклад, той же алгоритм RSA.

Крипостійкість даного методу визначається трудомісткістю обчислення дискретного логарифма:

f (x) = GX mod p,

де p - просте число, x - довільне натуральне число, g - деякий примітивний елемент поля Галуа GF (p). Загальновизнано, що інвертування функції GX mod p, тобто дискретного логарифмування, є важкою математичної завданням.

Сама процедура або протокол вироблення загального ключа полягає в наступному. Значення p і g є загальнодоступними параметрами протоколу. Абоненти А і В незалежно один від одного випадково вибирають по одному великому натуральному числу Ха і хВ. Це їх секретні ключі. Далі кожен з них обчислює відкриті ключі:

уа = gxA mod p і УВ = gxB mod p.

Потім вони обмінюються цими елементами по каналу зв'язку. Тепер абонент А, отримавши Yв і знаючи свій секретний елемент ХА, обчислює новий елемент:

уВхА mod p = (gxB) Xa mod p.

Аналогічно надходить абонент В:

уАхВ mod p = (gxA) xB mod p.

Тим самим у А і В з'явився спільний елемент поля, рівний gxA xB. Цей елемент і оголошується загальним ключем абонентів А і В.

Незворотність перетворення в цьому випадку забезпечується тим, що досить легко обчислити показову функцію в кінцевому поле Галуа, що складається з p елементів (p - просте число). Зворотній завдання обчислення x з y буде досить складною. Якщо p вибрано досить правильно, то витяг логарифма потребують обчислень, пропорційних L (p) = exp {(ln p ln ln p) 0.5}.

При всій простоті алгоритму Діффі-Хелмана його недоліком в порівнянні з

системою RSA є відсутність гарантованої нижньої оцінки трудомісткості розкриття ключа.

5.9 Алгоритми практичної реалізації криптосистем з відкритим ключем

5.9.1 Зведення в ступінь за модулем *m*

У криптографії використовуються обчислення за mod *m*, так як алфавіт будь-якої криптографічного системи є кінцева множина цілих чисел. Арифметика відрахувань до того ж легше реалізується на комп'ютерах, оскільки вона обмежує діапазон проміжних значень і результату. Для *k*-бітових відрахувань *m* проміжні результати будь-якого додавання, віднімання або множення будуть не довше, ніж 2*k* біт. Тому в арифметиці відрахувань ми можемо виконати зведення в ступінь без величезних проміжних результатів. Обчислення ступеня деякого числа по модулю іншого числа,

$a^d \bmod m$

є просто послідовність множень і поділок, але існують прийоми, що прискорюють цю дію. Один з таких прийомів прагне мінімізувати кількість множень по модулю, інший - оптимізувати окремі множення по модулю. Так як операції дистрибутивні, швидше виконати зведення в ступінь як потік послідовних множень, щоразу отримуючи відрахування.

Наприклад, для того, щоб обчислити $a^8 \bmod m$, не виконуйте сім множень і одне приведення по модулю; замість цього виконайте три менших множення і три менших приведення по модулю:

$a^8 \bmod m = ((a^2 \bmod m)^2 \bmod m)^2 \bmod m$

Точно також, $a^{16} \bmod m = (((a^2 \bmod m)^2 \bmod m)^2 \bmod m)^2 \bmod m$.

Обчислення $a^d \bmod m$, де *d* не є ступенем 2, не набагато важче. Двійковий запис являє *x* у вигляді суми ступенів 2: число 25 - це бінарне 11001, тому 25 = 16 + 8 + 1. Тоді:

$a^{25} \bmod m = (a\ a^8\ a^{16}) \bmod m = (a*(((a*a^2)\ ^2)\ ^2)\ ^2) \bmod m$

З продуманим збереженням проміжних результатів нам знадобиться тільки шість множень:

$((((((a^2 \bmod m)*a)^2 \bmod m)^2 \bmod m)^2 \bmod m)^2 \bmod m\ *a) \bmod m$

Такий прийом називається методом двійкових квадратів і множень. Він використовує простий і очевидний ланцюжок складань, в основі якої лежить двійкове уявлення числа. Збільшення швидкості обчислень при множенні 200-бітових чисел буде дуже помітним.

Алгоритм обчислення $a^d \bmod m$

Нехай натуральні числа *a* і *d* не перевищують за величиною *m*.

Уявімо *d* в двійковій системі числення:

$d = d_o 2^r + \ldots + d_{r-1} 2 + d_r$

де *r* - число двійкових розрядів, d_i - цифри в двійковому поданні, рівні 0 або 1, і $d_o = 1$.

Покладемо $a_o = a$ і потім для $i = 1, \ldots. r$ обчислимо:

$a_i \equiv a^2\ \ a^{di} \bmod m$

Тоді *ar* є бажаємий лишок $a^d \bmod m$.

Справедливість цього алгоритму випливає з легко доказуваного індукцією по *i* порівняння:

$a_i \equiv\ a^{do2i+\ldots+di} \bmod m$

Так як кожне обчислення на кроці 2 вимагає не більше трьох множень по модулю *m* і цей крок виконується $r \le \log_2 m$ раз, то складність алгоритму можна оцінити величиною 0 (ln *m*). Кажучи про складність алгоритмів, ми маємо на увазі кількість арифметичних

операцій.

Другий алгоритм - це класичний алгоритм Евкліда обчислення найбільшого загального дільника цілих чисел. Ми припускаємо заданими два натуральних числа a і b і обчислюємо їх найбільший спільний дільник НСД (a, b).

5.9.2 Алгоритм Евкліда обчислення НСД

Одним із способів обчислення НСД двох чисел є алгоритм Евкліда, який описав його з своїй книзі «Начала» близько 300 років до н.е. Вважають, що він не винайшов його, а сам алгоритм ще старше років на 200. Це найдавніший нетривіальний алгоритм, який актуальний і в наш час.

- Обчислимо r - залишок від ділення числа a на b ($a > b$):
  $a = b\,q + r,\ 0 \le r < b$
- Якщо $r = 0$, то b є шукане число.
- Якщо $r \neq 0$, то замінимо пару чисел (a, b) парою (b, r) і перейдемо до кроку 1.

Зупинка гарантується, оскільки залишки від поділів утворюють строго спадну послідовність.

Оцінку складності цього алгоритму дає наступна теорема.

Теорема. При обчисленні найбільшого загального дільника НЗД (a, b) за допомогою алгоритму Евкліда буде виконано не більше $5p$ операцій поділу із залишком, де p є кількість цифр в десяткового запису меншого з чисел a і b.

5.9.3 Обчислення зворотних величин в кільці цілих чисел

При обчисленні ключів в асиметричних криптографічних системах необхідно знаходити зворотні елементи в кільці цілих чисел Z_m. Елемент $x \in Z_m$ є зворотним до $a \in Z_m$, ($x = a^{-1}$), якщо

$x \cdot a \equiv 1 \bmod m$ или $a^{-1} \equiv x \bmod m$

У загальному випадку це порівняння може не мати рішень або мати кілька рішень. Воно має єдине рішення тоді і тільки тоді, якщо числа a і m взаємно прості, тобто НСД (a, m) = 1.

Розглянемо основні способи знаходження зворотних величин в Z_m.

- Метод перебору. Перевірити по черзі значення $x \in (1, 2, \ldots . m\text{-}1)$, поки не виконається порівняння $x \cdot a \equiv 1 \bmod m$.
- Якщо відома функція Ейлера (для RSA) $\phi(m)$, то можна обчислити $a^{-1} \bmod m \equiv a^{\phi(m)-1} \bmod m$, використовуючи алгоритм швидкого зведення в ступінь. (Це випливає з твердження теореми Ейлера: якщо a і m взаємно прості, то $a^{\varphi(m)} \equiv 1 \bmod m$.)
- Знаходження зворотної величини $a^{-1} \bmod m$ за допомогою розширеного алгоритму Евкліда.

Алгоритм Евкліда можна узагальнити способом, який має велике практичне значення. При цьому способі під час обчислення НСД можна водночас обчислити такі цілі числа x і y, що

$a \cdot x + b \cdot y =$ НСД (a, b)

Цей варіант називається розширеним алгоритмом Евкліда. Для обчислення зворотної величини використовується приватний режим роботи алгоритму Евкліда, при якому:

$b = m$ и НСД (a, m) = 1

Справді порівняння $x \cdot a \equiv 1 \bmod m$ означає, що $\exists$ ціле k, що вірно:

$a\,x + m\,k = 1 \bmod m$

Це завдання рівносильне пошуку цілих рішень рівняння
$ax + mk = 1$
Алгоритм пошуку цілих рішень рівняння a x + m k = 1
Трохи підправив алгоритм Евкліда, можна досить швидко вирішувати порівняння
$ax + mk = 1 \bmod m$
за умови, що НСД (a, m) = 1.

- Визначимо матрицю E = I - одинична матриця розміром 2x2.
- Обчислимо r - залишок від ділення числа a на m:
  $a = m\,q + r,\ 0 \le r < m$
- Якщо $r = 0$, то другий стовпець матриці E дає вектор $[x, k]^T$ рішень рівняння.
  - Якщо $r \neq 0$, то замінимо матрицю E матрицею

$$E = E \bullet \begin{pmatrix} 1 & 0 \\ 0 & -q \end{pmatrix}$$

  - Замінимо пару чисел (a, m) парою (m, r) і перейдемо до кроку 1.

Три наведених вище алгоритму відносяться до розряду так званих поліноміальних алгоритмів. Ця назва носить алгоритми, складність яких оцінюється зверху ступеневим чином в залежності від довжини записи вхідних чисел. Якщо найбільше з чисел, що подаються на вхід алгоритму, не перевищує m, то складність алгоритмів цього типу оцінюється величиною O ($ln^c m$), де c - деяка абсолютна постійна. У всіх наведених вище прикладах $c = 1$.

Поліноміальні алгоритми в теорії чисел - велика рідкість. Та й оцінки складності алгоритмів найчастіше спираються на будь-які недоведені, але правдоподібні гіпотези, зазвичай пов'язані з аналітичною теорією чисел.

5.9.4 Генерація простих чисел

Для алгоритмів з відкритими ключами потрібні прості числа. Їх потрібно безліч для будь-якої досить великої мережі, чи вистачить їх?

Так, існує приблизно 10^{151} простих чисел довжиною до 512 біт включно. Для чисел, близьких m, ймовірність того, що випадково вибране число виявиться простим, дорівнює $1/\ln m$. Тому повне число простих чисел, менших m, так само $m/(\ln m)$. При виборі з 10^{151} простих чисел ймовірність збігу вибору практично дорівнює нулю.

Але якщо так складно розкладання на множники, як може бути простим генерація простих чисел? Справа в тому, що відповісти "так" або "ні" на питання "чи є число простим?" набагато простіше, ніж відповісти на більш складне питання "які множники m?"

Генерація випадкових чисел з подальшою спробою розкладання їх на множники - це неправильний спосіб пошуку простих чисел. Існують різні імовірнісні перевірки на простоту чисел, що визначають, чи є число простим, із заданою ступенем достовірності. За умови, що ця «ступінь достовірності» досить велика, такі способи перевірки досить хороші.

Пропонуються наступні тести перевірки чисел на простоту:

Тест Леманна (Lehmann)

Ось послідовність дій при перевірці простоти числа p;

- Виберіть випадково число a, менше p.
- Розрахуйте $a^{(p-1)/2} \bmod p$.
- Якщо $a^{(p-1)/2} \neq 1$ або $-1 \bmod p$, то p не є простим.
- Якщо $a^{(p-1)/2} \equiv 1$ або $-1 \bmod p$, то ймовірність того, що число p не є простим, не більше ніж 50 відсотків.

Число a, яке не показує, що число p не є простим числом, називається свідком. Якщо p- складене число, ймовірність випадкового числа a бути свідком складовою природи числа p не менше 50 відсотків. Повторіть цю перевірку t раз з t різними значеннями a. Якщо результат обчислень дорівнює 1 або -1, але не завжди дорівнює 1, то p є простим числом з ймовірністю помилки $1/2^t$.

Алгоритм Рабіна-Міллера (Rabin-Miller)

Повсюди використовується простий алгоритм, розроблений М. Рабіном, частково заснованим на ідеях Міллера.

Виберіть для перевірки випадкове число p. Обчисліть b - число поділок p-1 на 2 (тобто $2b$ - це найбільша ступінь числа 2, на яку ділиться p-1). Потім обчисліть m, таке, що $p = 1 + 2b \cdot m$.

- Виберіть випадкове число a, менше p.
- Встановіть $j = 0$ і $z = a \cdot m \bmod p$.
- Якщо $z = 1$ або якщо $z = p$-1, то p проходить перевірку і може бути простим числом.
- Якщо $j > 0$ і Z = 1, то p не є простим числом.
- Встановіть $j = j + 1$. Якщо $j < b$ і $z < p$ - 1, встановіть $z = z^2 \bmod p$ і поверніться на етап (4). Якщо $z = p$ - 1, то p проходить перевірку і може бути простим числом.
- Якщо $j = b$ і $z \neq p$ - 1, то p не є простим числом.

У цьому тесті ймовірність проходження перевірки складовим числом спадає швидше, ніж в попередніх. Гарантується, що три чверті можливих значень a виявляться свідками. Це означає, що складене число прослизне через t перевірок з ймовірністю, що не більшою $1/4^t$, де t - число ітерацій. Насправді і ці оцінки дуже песимістичні. Для більшості випадкових чисел близько 99.9 відсотків можливих значень є свідками.

Існують більш точні оцінки. Для n-бітового кандидата в прості числа (де $n > 100$) ймовірність помилки в одному тесті менше, ніж 4^{n2}. І для 256-бітового n ймовірність помилки в шести тестах менше, ніж $\sqrt{7}^{2}$.

Практичні міркування

У реальних додатках генерація простих чисел відбувається швидко. Алгоритм наступний:

- Згенеруйте випадкове n-бітове число p.
- Встановіть старші і молодші біти рівними 1. (Старший біт гарантує необхідну довжину простого числа, а молодший біт забезпечує його непарність.)

- Переконайтеся, що p не ділиться на невеликі прості числа: 3, 5, 7, 11 і т.д. У багатьох реалізаціях перевіряється подільність p на всі прості числа, менші 256. Найбільш ефективною є перевірка на подільність для всіх простих чисел, менших 2000.

- Виконайте тест Рабіна-Міллера для деякого випадкового a. Якщо p проходить тест, згенеруйте інше випадкове a й повторіть перевірку. Обирайте невеликі значення a для прискорення обчислень. Виконайте п'ять тестів. (Одного може здатися достатнім, але виконайте п'ять.) Якщо p не проходить однієї з перевірок, згенеруйте інше p і спробуйте знову.

Інакше, можна не генерувати p випадковим чином кожен раз, але послідовно перебирати числа, починаючи з випадково обраного до тих пір, поки не буде знайдено просте число.

Етап 3 не є обов'язковим, але це хороша ідея. Перевірка, що випадкове непарне p не ділиться на 3, 5 і 7 відсікає 54% непарних чисел ще до етапу 4. Перевірка подільності на всі прості числа, менші 100, прибирає 76% непарних чисел, перевірка подільності на всі прості числа, менші 256, прибирає 80% непарних чисел. У загальному випадку, частка непарних кандидатів, які не діляться ні на одне просте число, менше m, дорівнює $1.12/1n$

Чим більше перевіряється m, тим більше попередніх обчислень потрібно виконати до тесту Рабіна-Міллера.

Сильні прості числа

Якщо m - добуток двох простих чисел p і q, то може знадобиться використовувати в якості і q *сильні прості числа*.

Такі прості числа мають ряд властивостей, які ускладнюють розкладання добутку m певними методами розкладання на множники. Серед таких властивостей були запропоновані:

- Найбільший спільний дільник p-1 і q-1 повинен бути невеликим.
- I p-1, і q-1 повинні мати серед своїх множників великі прості числа, відповідно p' і q'.
- I p'-1, і q'-1 повинні мати серед своїх множників великі прості числа.
- I p + 1, і q + 1 повинні мати серед своїх множників великі прості числа.
- I (p-1)/2, і (q-1)/2 повинні бути простими (зверніть увагу, при виконанні цієї умови виконуються і два перших).

Наскільки суттєво застосування саме сильних простих чисел залишається предметом триваючих суперечок. Ці властивості були розроблені, щоб затрудняти виконання ряду старих алгоритмів розкладання на множники. Однак найшвидші алгоритми однаково швидкі при розкладанні на множники будь-яких чисел, як задовольняють наведеним умовам, так і ні.

Деякі автори виступають проти спеціальної генерації сильних простих чисел, стверджуючи, що довжина простих чисел набагато важливіше їх структури. Більш того, сама структура зменшує випадковість числа і може знизити стійкість системи.

Але все може змінитися. Можуть бути створені нові методи розкладання на множники, які краще працюють з числами, що володіють певними властивостями. У цьому випадку знову можуть знадобитися сильні прості числа.

Лекція №6 ЕЛЕКТРОННИЙ ЦИФРОВИЙ ПІДПИС

В останні роки повсюдно простежується тенденція перекладу всього документообігу в електронну форму. Основна проблема, яка при цьому виникає - це встановлення автентичності автора і відсутності змін в отриманому документі тобто проблема аутентифікації як автора документа так і самого документа. У звичайному паперовому діловодстві ці проблеми вирішуються за рахунок того, що інформація в документі і рукописний підпис автора жорстко пов'язані з фізичним носієм, тобто папером. В електронних документах на машинних носіях такого зв'язку немає.

Принципово новим рішенням є електронний цифровий підпис (ЕЦП), який використовується для аутентифікації документа, переданого телекомунікаційними каналами. Функціонально він аналогічний звичайному рукописному підпису і володіє його основними перевагами:

- гарантує цілісність підписаного тексту, тобто відсутність виправлень і підчисток;
- засвідчує, що підписаний текст виходить від особи, яка його поставила;
- не дає цього самого обличчя можливості відмовитися від зобов'язань, пов'язаних з підписаним текстом.

Інакше кажучи, вона забезпечує аутентифікацію і цілісність повідомлень, тобто гарантує, що повідомлення надходять від достовірного відправника і в неперекрученому вигляді. Більш того, одержувач повідомлення не тільки переконується в його достовірності, а й отримує електронний підпис, який в подальшому може використовуватися як доказ достовірності повідомлення третім особам (арбітру) в тому випадку, якщо відправник згодом спробує відмовитися від свого підпису. Тут є повна аналогія звичайного підпису на папері, за винятком того, що під арбітром зазвичай розуміється технічний експерт, який дає висновок про достовірність електронного підпису, тобто виконує функцію, аналогічну функції графолога в разі звичайного підпису. Застосуванням схеми електронного підпису може бути наприклад платіжне доручення при безготівкових розрахунках з пластикових карт.

В якості підписаного документа може бути використаний будь-який файл. Підписаний файл створюється з непідписаного шляхом додавання в нього однієї або більше електронних підписів.

Кожен підпис містить наступну інформацію:

- дату підпису і термін закінчення дії ключа даного підпису;
- ідентифікатор, хто саме підписав (ім'я відкритого ключа);
- інформацію про особу, яка підписала файл;
- власне цифровий підпис.

6.1 Цифровий підпис на основі симетричних криптосистем

Для реалізації схеми цифрового підпису зазвичай вибирається будь-яка криптосистема, частіше асиметрична, але можна використовувати і симетричну. Такі криптосистеми використовуються рідше в протоколах цифрового підпису, так як вимагають залучення так званого посередника - абонента, який заслуговує на безумовну довіру всіх сторін. Розглянемо спочатку таку схему.

Нехай відправник - абонент А хоче підписати цифрове повідомлення і послати його адресату - абоненту В. Використовуючи симетричний шифр $E = \{E_k\}$, він може це зробити за допомогою посередника - абонента С, що має ключ k_1 для секретного зв'язку з відправником А, і ключ k_2 для секретного зв'язку з адресатом В. Ці ключі повинні бути встановлені заздалегідь до початку протоколу і можуть використовуватися багаторазово для декількох підписів.

Викладемо протокол по крокам.

1 Відправник А шифрує з використанням ключа k_1 повідомлення m для адресата В і відправляє криптограму c посереднику, де

$E_{k1}(m) = c$

2 Посередник розшифровує криптограму *c*, використовуючи ключ k_1: $E_{k1}^{-1}(c) = m$

3 Посередник С формує блок інформації $I_A = [m, c, p_A]$, що складається з розшифрованого повідомлення *m*, копії криптограми *c* і інформаційного блоку p_A, що підтверджує, що повідомлення виходило від А (ідентифікатор абонента А). Використовуючи ключ k_2, він шифрує блок I_A і посилає криптограму *d* одержувачу В, де $E_{k2}(I_A) = d$.

4 Абонент В розшифровує криптограму *d*, використовуючи ключ k_2: $E_{k2}^{-1}(d) = I_A = [m,c,p_A]$

Тепер він може прочитати повідомлення *m* та сертифікат посередника p_A про те, що його відправив А.

До переваг протоколу слід віднести наступне:

- Тільки посередник і відправник А спільно володіють секретним ключем k_1, тому тільки А міг відправити посереднику повідомлення, зашифроване ключем k_1 (підпис не підробляється), і підтвердження посередником цього підпису достовірно. Якщо активний противник спробує видати себе за відправника А, посередник виявить це на кроці 2 і не відправить повідомлення адресату В.
- Цей підпис не тиражується і підписаний документ не змінюється. Якби В спробував сертифікат посередника p_A об'єднати з неправдивим повідомленням *m′*, тобто заявити, що він отримав блок з повідомленням *m′* і сертифікатом p_A, то посередник виявив би цю невідповідність. Він запросив би у В це неправдиве повідомлення *m′* і криптограму *c*. Зашифрувавши неправдиве повідомлення *m′* на ключі k_1, посередник виявив би, що $E_{k1}(m') \neq c$, тобто обчислена криптограма не збігається з криптограмою *c*, яку йому надіслав В. Звичайно, В не міг би пред'явити криптограму *c′*, яка узгоджується з *m′*, тому що він не знає ключа k_1.
- Від цього підпису не можна відректися. Якщо відправник А пізніше заявить, що він не посилав повідомлення *m*, то те що зберігається у В сертифікат посередника p_A разом з повідомленням *m* засвідчать зворотне.

Таким чином, даний протокол усуває недоліки, властиві традиційному підпису на паперовому документі.

Розглянемо ще один протокол за участю посередника, завдання якого - забезпечити пред'явлення одержувачем В третій особі D підписаного документа *m*, отриманого ним від відправника А.

1) Абонент В об'єднує в блок $I = [m, p_A]$ отримане повідомлення *m* з сертифікатом посередника p_A і шифрує блок I на ключі k_2: $E_{k2}(I_A) = w$,

після цього В посилає криптограму *w* посереднику С (для пересилки третій особі D).

2) Посередник, використовуючи ключ k_2, розшифровує криптограму *w*: $E_{k2}^{-1}(w) = I$

3) Посередник С шифрує блок I на секретному ключі k_3, що використовується для зв'язку з D, і посилає D криптограму *v*: $E_{k3}(I) = v$.

4) D розшифровує криптограму *v*, використовуючи k_3: $E_{k3}^{-1}(v) = I = [m, p_A]$

Тепер В може прочитати і повідомлення *m*, і сертифікат посередника p_A, який свідчить, що його послав А.

Ці протоколи в принципі здійсненні, але вони вимагають високопродуктивної та безпомилкової роботи посередника. Він повинен постійно розшифровувати і зашифровувати повідомлення між кожною парою абонентів, які бажають відправити один одному підписаний документ. Посередник є вузьким місцем в такій системі зв'язку, навіть якщо він - бездушна комп'ютерна програма. Посередник повинен бути непогрішний, бо

навіть єдина помилка в мільйон підписів підриває довіру до нього. Крім того, він повинен працювати в цілковитій безпеці. Якщо його база даних по секретним ключам стане доступною зловмисникові або хтось спотворить його ключі, функціонування системи буде порушено. Цей протокол хороший швидше в теорії, ніж на практиці.

Щоб зловмисник не зміг скористатися багаторазово підписаним документом (наприклад, грошовим переказом), призначеним для разового використання, в документ слід додати інформаційний блок, де зафіксований час і дата підписання документа, так звана мітка часу. Крім того, мітка часу на документі знижує ризик відмови учасника від свого підпису (якщо той хто підписав заявить, що у нього вкрали закритий ключ і документ підписаний не їм). Відповідний протокол з посередником передбачає наступні кроки:

- Абонент А підписує повідомлення.

- Абонент А генерує заголовок, що містить таку нормативну інформацію, потім додає заголовок до підписаного повідомлення, підписує всі разом і відправляє посереднику.
- Посередник перевіряє справжність зовнішньої підписи і підтверджує таку нормативну інформацію. Далі він проставляє позначку часу на повідомленні і на ідентифікаційній інформації. Потім він підписує весь пакет і відсилає його як автору підпису А, так і адресату В.

- Абонент В перевіряє справжність підпису посередника, ідентифікаційну інформацію і справжність підпису А.

- Абонент А перевіряє справжність повідомлення, яке посередник послав В. Якщо А не визнає свого авторства, він негайно заявляє про це.

6.2 Цифровий підпис на основі асиметричних криптосистем

Розглянемо більш практичні протоколи цифрового підпису, що використовують асиметричні криптосистеми.

Взагалі кажучи, в криптосистемах з відкритим ключем в якості цифрового підпису може використовуватися результат шифрування документа на закритому ключі. Кожен, хто має відкритий ключ, може скористатися ним для розшифрування і тим самим перевірити достовірність підпису. Такий протокол відповідає всім вимогам цифрового підпису та не потребує посередника.

Але головна проблема полягає в тому, що підписання великих документів за допомогою їх шифрування асиметричним алгоритмом неефективно через невисоку швидкість шифрування. Тим більше, що в багатьох випадках сам документ в шифруванні не потребує, так як не містить секретної інформації, проте необхідно забезпечити його цілісність і підтвердження авторства. Тому, як правило, підписують шляхом асиметричного шифрування не сам документ, а отриману з нього згортку, звану хеш-функцією. Крім економії часу на постановку і перевірку підпису, в даному варіанті ще й зберігання підпису вимагає менше пам'яті і може здійснюватися окремо від зберігання документа.

Одержувачу відправляється сам документ і його зашифрована згортка як підпис під цим документом. Одержувач, володіючи тільки відкритим ключем, може перевірити підпис, але не може його підробити, так для створення підпису необхідний секретний ключ. При цьому важливим питанням є захист загальнодоступної бази відкритих ключів.

Відмінність схеми цифрового підпису від схеми передачі зашифрованої інформації (див. Рис. 5.1) при асиметричному шифруванні полягає в тому, що в цій схемі джерело ключів повинен бути перенесений на сторону відправника. Отже, цифровий підпис є відносно невелика кількість додаткової цифрової інформації, що передається разом з

підписаним текстом. Протокол електронного цифрового підпису (ЕЦП) включає в себе дві процедури:

- процедуру поставлення підпису; в ній використовується секретний ключ відправника повідомлення;

- процедуру перевірки підпису, в ній використовується відкритий ключ відправника.

При формуванні ЕЦП відправник насамперед обчислює хеш-функцію h (M) від підписаного тексту M. Обчислення значення хеш-функції $m = h$ (M) являє собою один короткий блок інформації m, що залежить від усього тексту M в цілому. Потім число m шифрується секретним ключем відправника і результат шифрування є ЕЦП для даного тексту M.

При перевірці ЕЦП одержувач повідомлення також обчислює хеш-функцію $m = h$ (M) прийнятого по каналу тексту M за відомим алгоритмом хешування. Після цього він розшифрував отриману разом з текстом підпис за допомогою відкритого ключа відправника і порівнює, чи відповідає отриманий результат обчисленому значенню m хеш-функції.

Неможливість підробки ЕЦП гарантується двома моментами: неможливістю визначити секретний ключ по відкритому в асиметричних криптосистемах і неможливістю зміни документа таким чином, щоб хеш-функція залишилася без зміни.

6.3 Хеш-функції

Хеш-функція призначена для стиснення підписаного документа M до декількох десятків або сотень біт. Значення хеш-функції h (M) складним чином залежить від документа M і не дозволяє відновити сам документ M. Хеш-функція $h(\cdot)$ приймає в якості аргументу повідомлення (документ, файл) M довільної довжини і повертає хеш-значення h (M) = m фіксованої довжини. Таким чином, інформація хешування є стислим поданням основного повідомлення довільної довжини. Іноді результат хешування називають дайджестом повідомлення.

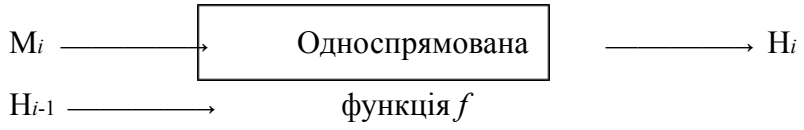
Хеш-функція повинна відповідати таким вимогам:

1) хеш-функція повинна бути однозначна;
2) хеш-функція повинна бути чутлива до всіляких змін в тексті M, таким, як вставки, викиди, перестановки і ін.
3) хеш-функція повинна мати властивість незворотності, тобто завдання підбору документа M', який володів би необхідним значенням хеш-функції, повинна бути обчислювально нерозв'язна;
4) ймовірність того, що значення хеш-функції двох різних документів (незалежно від їх довжин) співпадуть, повинна бути мізерно мала.

З визначення випливає, що для будь-якої хеш-функції є тексти-близнюки – які мають однакове значення хеш-функції, так як потужність множини аргументів необмежено більше потужності множини значень. Такий факт отримав назву "ефект дня народження".

Для побудови хеш-функцій, як правило, використовується одностороння функція f (·), Яка утворює вихідне значення довжиною n при завданні двох вхідних значень довжиною n.

Цими входами є блок вихідного тексту M_i і хеш-значення H_{i-1} попереднього блоку тексту: $H_i = f(M_i H_{i-1})$.



Хеш-значення, що обчислюється при введенні останнього блоку тексту, стає хеш-

значенням всього повідомлення М.

В результаті односпрямована хеш-функція завжди формує вихід фіксованої довжини незалежно від довжини вхідного тексту.

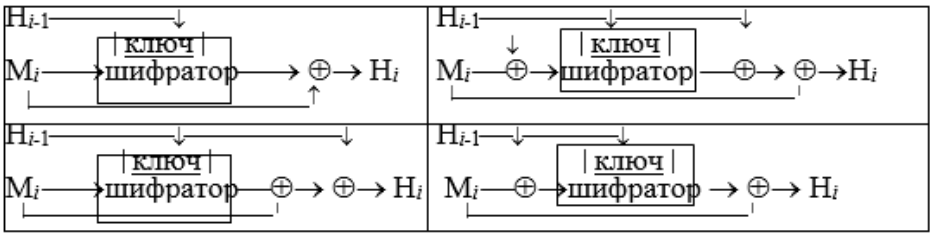
6.3.1 Хеш-функції на основі симетричних блокових алгоритмів

Односпрямованою хеш-функцію можна побудувати, використовуючи в якості опції $f(\cdot)$ симетричний блоковий алгоритм. Найбільш очевидний підхід полягає в тому, щоб шифрувати повідомлення М за допомогою блочного алгоритму в режимі CBC або CFB за допомогою фіксованого ключа і деякого вектора ініціалізації. Останній блок шифртекста можна розглядати в якості хеш-значення повідомлення М. При такому підході не завжди можливо побудувати безпечну односторонню хеш-функцію, але завжди можна отримати код аутентифікації повідомлення КАП.

Більш безпечний варіант хеш-функції можна отримати, використовуючи блок повідомлення в якості входу, попереднє значення - як ключ, а поточне хеш-значення - як вихід. Реальні хеш-функції проектуються ще більш складними. Довжина блоку зазвичай визначається довжиною ключа, а довжина хеш-значення збігається з довжиною блоку. Оскільки більшість блокових алгоритмів є 64-бітовими, схеми хешування проектують так, щоб хеш-значення мало довжину, рівну одинарній або подвійній довжині блоку.

Якщо прийняти, що отримувана хеш-функція коректна, безпеку схеми хешування базується на безпеці, що лежить в основі цього блочного алгоритму.

Нижче наведено чотири схеми безпечного хешування, у яких довжина хеш-значення дорівнює довжині блоку.



6.3.2 Алгоритм безпечного хешування SHA

Алгоритм безпечного хешування SHA (Secure Hash Algorithm) розроблений НІСТ і АНБ США в рамках стандарту безпечного хешування SHS (Secure Hash Standard) в 1992 р. Він розроблений для використання спільно з алгоритмом цифрового підпису DSA.

При введенні повідомлення М довільної довжини менше 2^{64} біт алгоритм SHA виробляє 160-бітове вихідне повідомлення, зване дайджестом повідомлення MD (Message Digest). Потім цей дайджест повідомлення використовується в якості входу алгоритму DSA, який обчислює цифровий підпис повідомлення М. Формування цифрового підпису для дайджесту повідомлення, а не для самого повідомлення підвищує ефективність процесу підписання, оскільки дайджест повідомлення зазвичай набагато коротший самого повідомлення. Такий же дайджест повідомлення повинен обчислюватися користувачем, перевіряючий отриманий підпис, при цьому в якості входу в алгоритм SHA використовується отримане повідомлення М.

Алгоритм SHA побудований на основі кінцевих автоматів Мілі. Головний цикл алгоритму обробляє по 512 біт повідомлення по черзі для всіх блоків, наявних в повідомленні. Він містить чотири цикли по 20 операцій кожен. Кожна операція реалізує нелінійну функцію, а потім виробляє зрушення і складання.

Алгоритм хешування SHA названий безпечним, тому що його творці стверджують, що обчислювально неможливо відновити повідомлення, відповідне даному дайджесту, а також знайти два різних повідомлення, які дадуть однаковий дайджест. Будь-яка зміна повідомлення при передачі з дуже великою ймовірністю спричинить зміну дайджесту, і

прийнятий цифровий підпис не пройде перевірку. Також вважається, що алгоритм SHA більш стійкий до атак повного перебору і атакам "дня народження", оскільки видає 160-бітове хеш-значення, тоді як більшість інших алгоритмів хешування формують 128-бітові хеш-значення.

6.4 Алгоритми електронного цифрового підпису

Технологія застосування системи ЕЦП передбачає наявність мережі абонентів, що посилають один одному підписані електронні документи. Для кожного абонента генерується пара ключів: секретний і відкритий. Секретний ключ зберігається абонентом в таємниці і використовується ним для формування ЕЦП. Відкритий ключ відомий всім користувачам і призначений для перевірки одержувачем дійсності підписаного електронного документа і автора підпису.

Для генерації пари ключів в алгоритмах ЕЦП, заснованих на асиметричних алгоритмах шифрування, використовуються математичні схеми, засновані на відомих складних обчислювальних завданнях: факторизації (розкладання на множники) великих цілих чисел і дискретного логарифмування.

Визначення. *Схема цифрового підпису* є конструкцією виду:
(K, M, S, P_K, V_K), де

- K - простір ключів, його елементи мають вигляд $k = (k_o, k_c)$, де
  k_o називається *відкритим ключем* (*public key*) або *відкритим компонентом ключа*,
  k_c називається *секретним ключем* (*secret key*) або *секретним компонентом ключа*;
- M - простір повідомлень;
- S - простір підписів;
- *функція побудови підпису* P_K: M → S
  ефективно будується по закритому ключу k_c;
- *функція перевірки підпису* V_K: {M x S} → {0; 1}
  ефективно будується по відкритому ключу k_o; і $\forall m \in M$ і $\forall s \in S$
  $V_K(m, s) = 1 <=> s = P_K(m)$;
  інакше $V_K(m, s) = 0$.
- не існує ефективного способу знайти без знання закритого ключа значення s для заданого m так, щоб $V_K(m, s) = 1$.

Остання вимога до цифрового підпису може бути посилено, а саме: потрібна неможливість знайти хоча б одну пару (m, s) таку, що:

$V_K(m, s) = 1$ («екзистенціальна підробка»)

Цифровий підпис дозволяє учаснику А передати учаснику В повідомлення в вигляді

$$\left(m, P_{K_A}\left(m\right)\right).$$

Тоді одержувач може бути впевнений, що повідомлення надіслано саме власником закритого ключа, і навіть довести це в суді.

Якщо цифровий підпис використовується спільно з шифруванням, то слід обчислювати підпис під відкритим текстом, а потім виконувати шифрування. Тобто передавати повідомлення у вигляді:

$E_K\left(m, P_K(m)\right)$, а не $\left(E_{K_B}(m), P_{K_A}\left(E_{K_B}(m)\right)\right)$

Підписувати шифртекст не рекомендується, так як при цьому не вдається зберігати розшифровані повідомлення разом з підписами.

Крім того, в цьому випадку від використовуваної криптосистеми потрібно стійкість до атак з відомим повідомленням. В іншому випадку, якщо по заданій шифрограмі c і заданому повідомленню m можна підібрати ключ k такий, що $E_k(m) = c$, зловмисник Z

може перехопити підпис $s = P_K\left(E_K(m)\right)$, для будь-якого повідомлення m' підібрати ключ

k такий, що $E_k(m') = E_K(m)$, зареєструвати цей ключ як свого і стверджувати, що А послав йому повідомлення c $s = P_A\left(E_k(m')\right)$.

6.4.1 Алгоритм цифрового підпису RSA

Першою і найбільш відомою у всьому світі конкретною системою ЕЦП стала система RSA. Узагальнена схема формування і перевірки цифрового підпису RSA показана на Рис. 5.1.

Криптосистему формує відправник (автор) електронних документів. Для цього обчислює два великих простих числа p і q, потім знаходить їх добуток $n = p \cdot q$ і значення функції Ейлера

$\phi(n) = (p-1)(q-1)$

Далі він обчислює пару ключів (секретний ключ d і відкритий ключ e) з умов:

$e \le \phi(n)$, НОД $(e, (\phi(n)) = 1$ і

$d < n$, $e \cdot d = 1 \bmod \phi(n)$

Пару чисел (e, n) автор передає партнерам по листуванню для перевірки його цифрових підписів. Число d зберігається автором як *секретний ключ* для підписування.

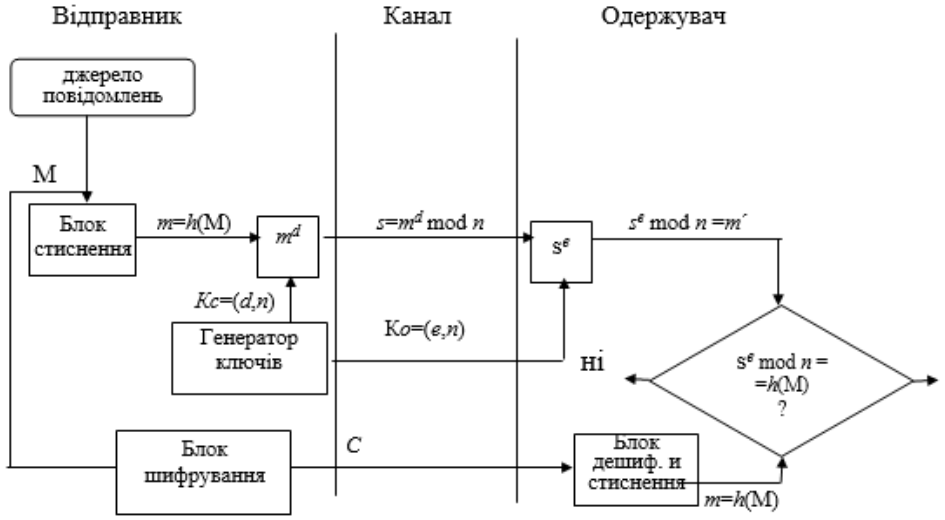


Рис. 6.1. Схема формування і перевірки цифрового підпису RSA

Припустимо, що відправник хоче підписати повідомлення M перед його відправкою. Спочатку повідомлення M (блок інформації, файл, ...) стискають за допомогою хеш-функції $h(\cdot)$ в ціле число $m = h$ (M). Потім обчислюють цифровий підпис s під електронним документом M, використовуючи хеш-значення m і секретний ключ d:

$$s = m^d \bmod n$$

Пара (M, s) передається одержувачу як електронний документ M, підписаний цифровим підписом s, причому підпис s сформований володарем секретного ключа d.

Після прийому пари (M, s) одержувач обчислює хеш-значення повідомлення M двома різними способами. Перш за все він відновлює хеш-значення m', застосовуючи криптографічне перетворення підпису s з використанням відкритого ключа e:

$$m' = s^e \bmod n$$

Крім того, він знаходить результат хешування прийнятого повідомлення M за допомогою такої ж хеш-функції $h(\cdot)$: $m = h$ (M).

Якщо дотримується рівність обчислених значень, тобто

$$s^e \bmod n = h \text{ (M)}$$

то одержувач визнає пару (M, s) справжньої. Доведено, що тільки власник секретного ключа d може сформувати цифровий підпис s по документу M, а визначити секретне число d з відкритого числу e не легше, ніж розкласти модуль n на множники.

Можна строго математично довести, що результат перевірки цифрового підпису s буде позитивним тільки в тому випадку, якщо при обчисленні s було використано секретний ключ d, відповідний відкритому ключу e. Тому s називають «ідентифікатором підписавшого».

Недоліки алгоритму цифрового підпису RSA

- При обчисленні модуля n, ключів e і d для системи цифрового підпису RSA необхідно перевіряти велику кількість додаткових умов, що зробити практично важко. Невиконання будь-якої з цих умов робить можливим фальсифікацію цифрового підпису з боку того, хто виявить таке невиконання. При підписанні важливих документів не можна допускати таку можливість навіть теоретично.
- Для забезпечення криптостійкості цифрового підпису RSA по відношенню до спроб фальсифікації на рівні, наприклад, національного стандарту США, необхідно використовувати при обчисленнях n, d і e цілі числа не менше 2^{512} (або близько 10^{154}) кожне, що вимагає великих обчислювальних витрат, що перевищують на 20-30%
- обчислювальні витрати інших алгоритмів цифрового підпису при збереженні того ж рівня криптостійкості.

- Цифровий підпис RSA уразливий до так званої мультиплікативної атаки. Інакше кажучи, алгоритм цифрового підпису RSA дозволяє зловмисникові без знання секретного ключа *d* сформувати підписи під тими документами, у яких результат хешування можна обчислити як добуток результатів хешування вже підписаних документів.

6.4.2 Алгоритм цифрового підпису Ель Гамаля (EGSA)

Більш надійний і зручний для реалізації на персональних комп'ютерах алгоритм цифрового підпису був розроблений в 1984 році американцем арабського походження Ель Гамалем. У 1991р. НІСТ США обґрунтував перед комісією Конгресса США вибір алгоритму цифрового підпису Ель Гамаля в якості основи для національного стандарту.

Ідея EGSA (El Gamal Signature Algorithm) заснована на тому, що для обґрунтування практичної неможливості фальсифікації цифрового підпису може бути використана більш складна обчислювальна задача, ніж розкладання на множники великого цілого числа - завдання дискретного логарифмування. Крім того, Ель Гамалю вдалося уникнути очевидній слабкості алгоритму цифрового підпису RSA, пов'язаної з можливістю підробки цифрового підпису під деякими повідомленнями без визначення секретного ключа.

Алгоритм цифрового підпису Ель Гамаля:

- Вибирають *несекретні параметри* алгоритму: два великих цілих числа p - просте і $g < p$: p (~ 10^{308} або ~ 2^{1024}) і g (~ 10^{154} або ~ 2^{512});

- Відправник вибирає *секретний ключ* x для підписування документів. Їм є випадкове ціле число $x \in Z_p$, тобто $1 < x < p$;

- Відправник обчислює *відкритий ключ* y, який використовується для перевірки свого підпису. Він і обчислює:
  $y = g^x \bmod p$
  і передає його всім потенційним одержувачам документів;

- Відправник хешує повідомлення M за допомогою хеш-функції $h(\cdot)$ в ціле число m:
  $m = h(M)$, $1 < m < (p\text{-}1)$

- Відправник генерує секретний параметр алгоритму: випадкове ціле число k, $1 < k < (p\text{-}1)$, таке, що k і $(p\text{-}1)$ є взаємно простими;

- Нарешті відправник обчислює цифровий підпис S = (a, b), що складається з двох цілих чисел a і b, що обчислюються наступним чином:
  $a = g^k \bmod p$
  і число b обчислюється за допомогою секретного ключа x з рівняння:
  $m = x \cdot a + k \cdot b \bmod (p - 1)$
  використовуючи розширений алгоритм Евкліда.

Трійка чисел (M, a, b) передається одержувачу, в той час як пара чисел (x, k) тримається в секреті.

Після прийому підписаного повідомлення (M, a, b) одержувач повинен перевірити, чи відповідає підпис S = (a, b) з повідомленням M. Для цього одержувач:

1. Хешує прийняте повідомлення M, тобто обчислює $m = h$ (M).

2. Обчислює значення $A = g^m \bmod p$.

3. Визнає повідомлення M справжнім тільки, якщо воно співпало з: $A = y^a a^b \bmod p$
   Інакше кажучи, він перевіряє справедливість співвідношення:

$A = y^a a^b \bmod p = g^m \bmod p$

Можна строго математично довести, що остання рівність буде виконуватися тоді і тільки тоді, коли підпис S = (*a*, *b*) під документом M отриманий за допомогою саме того секретного ключа *x*, з якого був отриманий відкритий ключ *y*. Таким чином, можна надійно упевнитися, що відправником повідомлення M був володар саме даного секретного ключа *x*, не розкриваючи при цьому сам ключ, і що відправник підписав саме цей конкретний документ M.

Слід зазначити, що виконання кожного підпису за методом Ель Гамаля вимагає нового значення *k*, причому це значення повинно вибиратися випадковим чином. Якщо порушник розкриє коли-небудь значення *k*, повторно використовуване відправником, то він зможе розкрити секретний ключ *x* відправника.

Схема Ель Гамаля є характерним прикладом підходу, який допускає пересилання повідомлення M у відкритій формі разом з приєднаним аутентифікатором (*a*, *b*). У таких випадках процедура встановлення автентичності прийнятого повідомлення складається в перевірці відповідності аутентифікатора повідомлення.

Приклад. Виберемо: числа $p = 11$, $g = 2$ і секретний ключ $x = 8$. Обчислюємо значення відкритого ключа:

$y = g^x \bmod p = y = 2^8 \bmod 11 = 3$

Припустимо, що вихідне повідомлення M характеризується хеш-значенням $m = 5$. Для того щоб обчислити цифровий підпис для повідомлення M, що має хеш-значення $m = 5$, спочатку виберемо випадкове ціле число $k = 9$.

Переконаємося, що числа *k* і (*p*-1) є взаємно простими. Дійсно, НСД (9,10) = 1.

Далі обчислюємо елементи *a* і *b* підпису:

$a = g^k \bmod p = 2^9 \bmod 11 = 6$

елемент *b* визначаємо, використовуючи розширений алгоритм Евкліда:

$m = x \cdot a + k \cdot b \bmod (p\text{-}1)$

При $m = 5$, $a = 6$, $x = 8$, $k = 9$, $p = 11$ отримуємо $b = 3$ як рішення рівняння

$5 = (6 \cdot 8 + 9 \cdot b) \bmod 10$ або $9 \cdot b \equiv -43 \bmod 10$

Отже, цифровий підпис представляє собою пару: $a = 6$, $b = 3$.

Далі відправник передає підписане повідомлення. Прийнявши підписане повідомлення і відкритий ключ $y = 3$, одержувач обчислює хеш-значення для повідомлення M: $m = 5$, а потім обчислює два числа:

$y^a a^b \bmod p = 3^6 \cdot 6^3 \bmod 11 = 10 \bmod 11$

$g^m \bmod p = 2^5 \bmod 11 = 10 \bmod 11$

Так як ці два цілих числа рівні, прийняте одержувачем повідомлення визнається справжнім.

Схема цифрового підпису Ель Гамаля має ряд переваг в порівнянні зі схемою цифрового підпису RSA:

- При заданому рівні стійкості алгоритму цифрового підпису цілі числа, які беруть участь в обчисленнях, мають запис на 25% коротший, що зменшує складність обчислень майже в два рази і дозволяє помітно скоротити обсяг використовуваної пам'яті.

- При виборі модуля *p* досить перевірити, що це число є простим і що у числа (*p*-1) є великий простий множник (тобто всього дві умови, що досить просто перевіряються).

- Процедура формування підпису за схемою Ель Гамаля не дозволяє обчислювати цифрові підписи під новими повідомленнями без знання секретного ключа (як в RSA).

Однак алгоритм цифрового підпису Ель Гамаля має і деякі, *недоліки* в порівнянні зі

схемою підпису RSA. Зокрема, довжина цифрового підпису виходить в 1,5 рази більшою, що в свою чергу збільшує час її обчислення.

6.4.3 Алгоритм цифрового підпису DSA

Алгоритм цифрового підпису DSA (Digital Signature Algorithm) запропонований в 1991 р. в НІСТ США для використання в стандарті цифрового підпису DSS (Digital Signature Standard). Алгоритм DSA є розвитком алгоритмів цифрового підпису Ель Гамаля і К.Шнорра. Він складається з наступних кроків:

- *Відправник і одержувач* електронного документа генерують використовувані при обчисленнях великі цілі числа g, p, q, що є відкритими параметрами криптосистеми:
  - g і p - прості числа, L біт кожне ($512 < L \le 1024$);
  - q - просте число довжиною 160 біт (дільник числа (p -1)).

  Вони можуть бути загальними для всіх користувачів мережі.

- *Відправник* вибирає випадкове ціле число x ($1 < x < q$), що є його секретним ключем.

- Потім відправник обчислює значення відкритого ключа:

  $y = g^x \bmod p$

  яке передається всім одержувачам документів.

- Для того щоб підписати документ M, відправник хешує його в ціле хеш- значення m з використанням односторонньої функції хешування $h(\cdot)$, визначеної в алгоритмі безпечного хешування SHA:

  $m = h(M)$, $1 < m < q$

  - Потім відправник генерує випадкове ціле число k, $1 < k < q$, і обчислює:

  $r = (g^k \bmod p) \bmod q$

  - Далі відправник обчислює за допомогою секретного ключа x число:

  $s = (m + r*x)/k \bmod q$

  Пара чисел (r, s) = S утворює цифровий підпис S під документом M.

  Таким чином, підписане повідомлення являє собою трійку чисел [M, r, s].

  - Одержувач підписаного повідомлення [M, r, s] перевіряє виконання умов $0 < r < q$, $0 < s < q$ і відкидає підпис, якщо хоча б одна з цих умов не виконана.

  - Потім одержувач обчислює хеш-функцію $m = h$ (M), значення:

  $w = 1/s \bmod q$

  і числа $U_1 = (m * w) \bmod q$

  $U_2 = (r * w) \bmod q$

  - Далі одержувач за допомогою відкритого ключа y обчислює значення

  $v = ((g^{U1} * y^{U2}) \bmod p) \bmod q$

  і перевіряє виконання умови $v = r$.

Якщо умова $v = r$ виконується, тоді підпис S = (r, s) під документом M визнається одержувачем справжнім.

Можна строго математично довести, що остання рівність буде виконуватися тоді і тільки тоді, коли підпис S = (r, s) під документом M отриманий за допомогою саме того секретного ключа x, з якого був отриманий секретний ключ y.

Таким чином, можна надійно упевнитися, що відправник повідомлення володіє саме даними секретним ключем x (не розплющуючи при цьому значення ключа x) і що відправник підписав саме даний документ M.

У порівнянні з алгоритмом цифрового підпису Ель Гамаля алгоритм DSA має наступні основні *переваги*:

- При будь-якому допустимому рівні стійкості, тобто при будь-якій парі чисел g і p (від 512 до 1024 біт), числа q, x, r, s мають довжину по 160 біт, скорочуючи довжину підпису до 320 біт.

- Більшість операцій з числами як при обчисленні підписи, так і при її перевірці проводиться по модулю числа q довжиною 160 біт, що скорочує обсяг пам'яті і час обчислення підпису.

Недоліком алгоритму DSA є те, що при підписуванні і при перевірці підпису доводиться виконувати складні операції ділення по модулю q:

$s = (m + r\,x)/k \bmod q$, $w = 1/s \bmod q$

що не дозволяє отримувати максимальну швидкодію.

Але цей недолік може бути усунутий за допомогою виконання попередніх обчислень. Зауважимо, що значення r не залежить від повідомлення M і його хеш-значення m. Можна заздалегідь створити рядок випадкових значень k і потім для кожного з цих значень обчислити значення r. Можна також заздалегідь обчислити зворотні значення k-1 для кожного із значень k. Потім, під час вступу повідомлення M можна обчислити значення s для даних значень r і k-1. Ці попередні обчислення значно прискорюють роботу алгоритму DSA.

Лекція №7 КРИПТОГРАФІЧНІ ПРОТОКОЛИ

7.1 Специфіка взаємодії віддалених абонентів

Успіхи, досягнуті в розробці схем цифрового підпису та відкритого розподілу ключів, дозволили застосувати ці ідеї також і до інших місій взаємодії віддалених абонентів. Так виник великий новий напрямок теоретичної криптографії - криптографічні протоколи.

У класичній шенновській моделі системи секретного зв'язку є два повністю довіряючих один одному учасника, яким необхідно передавати між собою інформацію, не призначену для третіх осіб. Вирішується завдання захисту секретної інформації від зовнішнього противника.

Об'єктом вивчення теорії криптографічних протоколів є віддалені абоненти, які взаємодіють, як правило, по відкритим каналам зв'язку. Мета взаємодії абонентів полягає у вирішенні конкретного завдання. У цій ситуації противником може виявитися будь-який з абонентів або кілька абонентів, що вступили в змову, і переслідують власні цілі. При цьому противник в різних завданнях може мати різні можливості: наприклад, може взаємодіяти з абонентами від імені інших абонентів або втручатись в обміни інформацією між абонентами та т. п. Тому криптографічні протоколи повинні захищати їх учасників не тільки від зовнішнього противника, а й від нечесних дій партнерів.

Є відмінності криптографічних протоколів від криптосистем, з яких можна виділити наступне:

- учасники протоколу, взагалі кажучи, не довіряють один одному;
- в протоколі може бути більше двох учасників;
- протоколи можуть бути інтерактивними, тобто припускати багатораундовий обмін повідомленнями між учасниками.

На жаль, поняття криптографічного протоколу, мабуть, неможливо формалізувати.

Визначення. Під протоколом (не обов'язково криптографічним) зазвичай розуміють розподілений алгоритм, що включає в себе:

- сукупність алгоритмів для кожного з учасників,
- специфікації форматів повідомлень, що пересилаються між учасниками,

- специфікації синхронізації дій учасників,
- опис дій при виникненні збоїв.

На останній елемент цього списку слід звернути особливу увагу, оскільки його часто не беруть до уваги, а некоректний повторний пуск може повністю зруйнувати безпеку учасників навіть в стійкому криптографічному протоколі.

Всі типи криптографічних протоколів можна умовно розділити на дві групи: прикладні протоколи і примітивні.

Прикладний протокол вирішує конкретну задачу, яка виникає (або може виникнути) на практиці.

Примітивні протоколи використовуються як своєрідні «будівельні блоки» при розробці прикладних протоколів.

За останнє десятиліття криптографічні протоколи перетворилися в основний об'єкт досліджень в теоретичній криптографії. Одна з основних областей додатків криптографічних протоколів - банківські платіжні системи, де замість платіжних доручень на папері використовується їх електронна форма. Вигоди від такої заміни настільки відчутні, що, мабуть, банки від неї вже ніколи не відмовляться, які б технічні та криптографічні труднощі при цьому не виникали. Але платіжні доручення - лише один з численних типів документів, що знаходяться в обороті в сфері бізнесу. Але ж існують ще документи, з якими працюють державні органи і громадські організації, юридичні документи і т. п. В останні роки чітко простежується тенденція перекладу всього документообігу в електронну форму.

У дослідженнях багатьох типів криптографічних протоколів зроблені тільки перші кроки і ще багато математичні проблеми треба буде розв'язати, перш ніж криптографічні протоколи увійдуть в повсюдне використання.

7.2 *Приклади криптографічних протоколів*

Познайомимося з деякими типами криптографічних протоколів, а також довкруги математичних задач, що виникають при дослідженні їх стійкості. Розглянемо кілька прикладів завдань, що вирішуються віддаленими абонентами.

1 *Протокол підписання контракту*. Взаємодіють два, які не довіряють один одному абонента. Вони хочуть підписати контракт. Це треба зробити так, щоб не допустити таку ситуацію: один з абонентів отримав підпис іншого, а сам не підписався.

2 *Протокол ідентифікації абонента*. Взаємодіють два абонента А і В. Абонент А хоче довести абоненту В, що він саме А, а не противник. Типовий приклад: А - власник інтелектуальної смарт-карти (кредитної картки, карти доступу в закрите приміщення, комп'ютерний ключ), В - банк, комп'ютер охорони, адміністратор мережі.

3 *Протокол візантійської угоди*. Взаємодіють кілька віддалених абонентів, які отримали накази з одного центру. Частина абонентів, включаючи центр, можуть бути противниками. Необхідно виробити єдину стратегію дій, виграшну для абонентів.

Це завдання прийнято називати завданням про візантійських генералів. Наведемо приклад, з яким ця задача зобов'язана своєю назвою. Візантія. Ніч перед великою битвою. Візантійська армія складається з n легіонів, кожен з яких підпорядковується своєму генералу. Крім того, у армії є головнокомандувач, який керує генералами. Однак імперія перебуває в занепаді і до однієї третини генералів, включаючи головнокомандувача, можуть бути зрадниками. Протягом ночі кожен з генералів отримує від головнокомандувача наказ про дії на ранок, причому можливі два варіанти наказу: «атакувати» або «відступати». Якщо всі чесні генерали атакують, то вони перемагають. Якщо всі вони відступають, то їм вдається зберегти армію. Але якщо частина з них атакує, а частина відступає, то вони зазнають поразки. Якщо головнокомандувач виявиться

зрадником, то він може дати різним генералам різні накази, тому накази головнокомандуючого не варто виконувати беззаперечно. Якщо кожен генерал буде діяти незалежно від інших, результати можуть виявитися сумними. Очевидно, що генерали потребують в обміні інформацією один з одним (щодо отриманих наказів) з тим, щоб прийти до угоди.

4 *Протокол підкидання монети по телефону*. Взаємодіють два, які не довіряють один одному абонента. Вони хочуть кинути жереб за допомогою монети. Це треба зробити так, щоб абонент, який підкидає монету, не міг змінити результат підкидання після отримання здогадки від абонента, який вгадує цей результат.

Наведемо один з найпростіших протоколів підкидання монети по телефону (так звана схема Блюма-Мікалі). Для його реалізації у абонентів A і B має бути одностороння функція f: X —> Y, яка задовольняє таким умовам:

- X - кінцева множина цілих чисел, яка містить однакову кількість парних і непарних чисел.
- будь-які числа $x_1, x_2 \in X$, мають один образ $f(x_1) = f(x_2)$, мають одну парність;
- по заданому образу $f(x)$ «важко» обчислити парність невідомого аргументу x.

Роль підкидання монети грає випадковий і рівноймовірний вибір елемента $x \in X$, а роль орла і решки - парність і непарність x відповідно. Нехай A - абонент, підкидає монету, а B - абонент, що вгадує результат. Протокол складається з наступних кроків:

- A вибирає x («підкидає монету»), зашифровує x, тобто обчислює $y = f(x)$, і посилає y абоненту B;
- B отримує y, намагається вгадати парність x і посилає свою здогадку абоненту A;
- A отримує здогадку від B і повідомляє B, вгадав чи він, посилаючи йому вибране число x;
- B перевіряє, чи не обманює A, обчислюючи значення $f(x)$ і порівнюючи його з отриманим на другому кроці значенням y.

7.3 Інтерактивна система доказу

Осмислення різних протоколів і методів їх побудови призвело в 1985-1986 рр. до появи двох плідних математичних моделей - *інтерактивної системи доказів* і *доказів з нульовим розголошенням*. Математичні дослідження цих нових об'єктів дозволили довести багато тверджень, досить корисних при розробці криптографічних протоколів.

Визначення. Під *інтерактивною системою доказу* (P, V, S) розуміють протокол взаємодії двох абонентів:

P (prover) – який доводить і

V (verifier) – який перевіряє.

Абонент P хоче довести абоненту V, що твердження S істинно. При цьому абонент V самостійно, без допомоги абонента P, не може перевірити твердження S (тому V і називається перевіряючим). Абонент P може бути і противником, який хоче довести абоненту V, що твердження S істинно, хоча насправді воно помилкове. Протокол може складатися з багатьох раундів обміну повідомленнями між абонентами P і V і повинен задовольняти двом умовам:

1 *повнота* - якщо S дійсно істинно, то абонент P переконає абонента V визнати це;

2 *коректність* - якщо S помилково, то абонент P «навряд чи» переконає абонента V, що S істинно.

Тут словами «навряд чи» ми для простоти замінили точне математичне формулювання.

Підкреслимо, що у визначенні системи (P, V, S) не допускалось, що абонент V може бути противником. А якщо це так і перевіряючий хоче «випитати», у того хто доводить якусь нову корисну для себе інформацію про затвердження S? В цьому випадку абонент P, звичайно, може не хотіти, щоб це сталося в результаті роботи даного протоколу. Протокол (P, V, S), який вирішує таке завдання, називається *доказом з нульовим розголошенням* і повинен задовольняти, крім умовам 1) і 2), ще й наступній умові:

3) *нульове розголошення* - в результаті роботи протоколу (P, V, S) абонент V не збільшить свої знання про затвердження S або, іншими словами, не зможе випитати ніякої інформації про те, чому S істинно.

7.4 Протоколи аутентифікації

Одне із застосувань інтерактивних систем доказу - протоколи аутентифікації, коли один з абонентів повинен довести іншому свій статус, але при цьому зробити так, щоб ніхто інший, який спостерігає цей протокол, не зміг зробити те ж саме. Такий, один з найбільш важливих і поширених типів криптографічних протоколів називають схемами аутентифікації.

Прикладом протоколу аутентифікації може служити всім відома казка про вовка і семеро козенят. У цьому протоколі коза виступає в якості того, хто доводить, а семеро козенят - як того, хто перевіряє. Протокол покликаний забезпечувати цілісність. В даному випадку - сімох козенят. У казці описана атака на протокол. Противником виступає вовк, і цей противник активний: спочатку він підслуховує виконання протоколу, потім намагається сам пройти аутентифікацію в якості того, хто доводить (кози) і при цьому накопичує й аналізує отриману інформацію. Протокол виявився нестійким проти активного противника.

Більш серйозне призначення і суть протоколів аутентифікації (або ідентифікації) легко зрозуміти на наступному прикладі. Уявімо собі інформаційну систему, яка працює в комп'ютерній мережі і забезпечує доступ до деяких даних. У адміністратора системи є список всіх її користувачів разом з співставленим кожному з них набором повноважень, на основі яких здійснюється розмежування доступу до ресурсів системи. Ресурсами можуть бути, наприклад, деякі фрагменти інформації, а також функції, які виконуються системою. Одним користувачам може бути дозволено читати одну частину інформації, іншим - іншу її частину, а третім - ще й вносити в неї зміни. В даному контексті під забезпеченням цілісності розуміється запобігання доступу до системи осіб, які не є її користувачами, а також запобігання доступу користувачів до тих ресурсів, на які у них немає повноважень. Найбільш поширений метод розмежування доступу - парольний захист не забезпечує достатньої стійкості. Тому перейдемо до криптографічній постановці завдання.

Для того, щоб протокол аутентифікації був стійким, досить, щоб він був доказом з нульовим розголошенням. У протоколі є два учасники - Аліса, яка повинна довести свою автентичність, і Боб, який цю автентичність повинен перевірити. У Аліси є два ключі - загальнодоступний відкритий K_o і секретний K_c. Фактично Алісі потрібно довести, що вона знає - K_c, і зробити це таким чином, щоб цей доказ можна було перевірити, знаючи тільки K_o.

Схема аутентифікації Шнорра

Одним з найбільш ефективних практичних протоколів аутентифікації вважається протокол Шнорра.

Нехай

1) p і q - прості числа такі, що q ділить p-1.

2) (Шнорр пропонує використовувати p довжини порядку 512 біт і q - довжини близько 140 бітів.)

3) $g \in Z_p$ таке, що $g^q = 1 \bmod p$, $g \neq 1$.

4) $x \in Z_q$ - секретний ключ

5) $y = g^x \bmod p$ - відкритий ключ.

Визначення x по y - це завдання дискретного логарифмування, для якої на даний момент не відомі ефективні алгоритми.

Відкриті ключі всіх учасників схеми повинні публікуватися таким чином, щоб виключалася можливість їх підміни (таке сховище ключів називається загальнодоступним сертифікованим довідником). Ця проблема, звана часто проблемою автентичності відкритих ключів, становить окремий предмет досліджень в криптографії.

У протоколі Шнорра кількість раундів дорівнює трьом. Наведемо алгоритм протоколу по крокам:

2 В якості секретного ключа схеми аутентифікації Аліса вибирає випадкове число
k з множини $\{1, ..., q - 1\}$, обчислює відкритий ключ $r = g^k \bmod p$ і посилає r Бобу.

3 Боб вибирає випадковий запит e з множини $\{0, ..., 2t - 1\}$, де t - деякий параметр, і посилає e Алісі.

4 Аліса обчислює $s = k + xe \bmod q$ і посилає s Бобу.

Боб перевіряє співвідношення $r = g^s y^e \bmod p$ і, якщо воно виконується, приймає доказ, в іншому випадку - відкидає.

Перше з вимог до стійкості протоколів аутентифікації - *коректність*, означає, що противник, знає тільки відкритий ключ y, може пройти аутентифікацію лише з зневажливо малою ймовірністю.

Нескладний аналіз показує, що коректність протоколу Шнорра залежить від обраного значення параметра t. Справді, якщо t невелика, то противник має хороші шанси просто вгадати той запит e, який він отримає від Боба на кроці 2.

Нехай для простоти $t = 1$. Тоді противник, який не знає секретного ключа x, може діяти таким чином. Підкинувши монету, він вибирає рівноймовірним чином одне зі значень 0 або 1. Позначимо його через e'. Далі противник вибирає довільне s з $\{0, ..., q\text{-}1\}$, обчислює $r = g^s y^{e'} \bmod p$ посилає r Бобу. Ясно, що запит e, отриманий від Боба на кроці 2, співпаде з e' з ймовірністю 1/2, і саме з такою ймовірністю противник пройде аутентифікацію. Якщо ж значення t досить велике, то шанси вгадати запит e малі. Шнорр рекомендує $t = 72$. Імовірність простого вгадування буде 2^{-72}, її можна вважати зневажливо малою.

Якщо в схемі Шнорра Аліса є противником, то на кроці 1 замість дій, запропонованих протоколом, вона може вибирати r довільним (але ефективним) чином. Іншими словами, Аліса використовує певний поліноміальний імовірнісний алгоритм, який для кожного конкретного значення r визначає ймовірність його вибору.

Нехай r - деяке значення, яке Аліса передала Бобу на кроці 1. Припустимо, що нам вдалося знайти два запити:

$e_1, e_2 \in \{0,...,2^t\text{-}1\}$, $e_1 \neq e_2$

такі, що Аліса може для кожного з них знайти відповідні значення s, для яких перевірка на кроці 4 дасть позитивний результат. Позначимо ці значення s через s_1 і s_2 відповідно. Ми маємо:

$r = g^{s1}y^{e1} \bmod p$

$r = g^{s2}y^{e2} \bmod p$

Звідси:

$(g^{s1}y^{e1} = g^{s2}y^{e2}) \bmod p$, або $(g^{s1-s2} = g^{e2-e1}) \bmod p$

Оскільки $e_1 \neq e_2$, існує $(e_1 - e_2)^{-1} \bmod q$ і, отже,

$(s_1-s_2)(e_2-e_1)^{-1} \bmod q = x$ - дискретний логарифм y.

Таким чином, або запити $e_1 \neq e_2$, такі, що Аліса може відповісти належним чином на обидва з них (при одному і тому ж r) на кроці 3 протоколи, зустрічаються «досить рідко», і це означає, що атака Аліси успішна лише зневажливо малою ймовірністю, або такі значення трапляються «досить часто», і тоді той алгоритм, який застосовує Аліса, можна використовувати для обчислення дискретних логарифмів.

Ця неформально викладена ідея була використана Шнорром для доказу поліноміальної звідності завдання дискретного логарифмування до завдання, що стоїть перед пасивним противником, тобто таким, який намагається пройти аутентифікацію, знаючи лише відкритий ключ. Іншими словами, доведено, що в припущенні труднощі завдання дискретного логарифмування схема аутентифікації Шнорра є стійкою проти пасивного противника, тобто коректною.

Активний противник може провести кілька сеансів виконання протоколу в якості перевіряючого з чесним який доводить (або підслухати такі виконання) і після цього спробувати атакувати схему аутентифікації. Для стійкості проти активного противника досить, щоб протокол аутентифікації був доказом з нульовим розголошенням. Однак властивість нульового розголошення для схеми Шнорра досі нікому довести не вдалося. Більш того, на даний момент відомий єдиний метод доказу властивості нульового розголошення - так званий метод «чорного ящика». У цьому методі машина, яка моделює використовує алгоритм перевіряючого лише в якості оракула, тобто, не аналізуючи сам цей алгоритм, подає йому на вхід будь-які значення за своїм вибором і отримує відповідні вихідні значення.

Доведено, що трьохраундові докази з нульовим розголошенням, в яких остання властивість встановлюється методом «чорного ящика», існують лише в тривіальному випадку, тобто коли перевіряючий може самостійно, без будь-якої допомоги, того хто доводить, перевірити істинність затвердженого. Відносно схеми Шнорра з цього результату випливає, що або існує ефективний алгоритм дискретного логарифмування, або властивість нульового розголошення цього протоколу не може бути доведено методом «чорного ящика». Питання про існування доказів з нульовим розголошенням, для яких властивість нульового розголошення не може бути доведено методом «чорного ящика», залишається відкритим.

Неважко показати, що схема Шнорра володіє трохи більш слабкою властивістю - властивістю нульового розголошення щодо чесного перевіряючого. У цьому випадку досить побудувати моделюючу машину тільки для чесного перевіряючого, який на кроці 2 і справді вибирає випадковий запит e з множини $\{0, ..., 2t\text{-}1\}$. Властивості нульового розголошення щодо чесного перевіряючого може виявитися достатнім, якщо схема аутентифікації використовується, наприклад, для контролю за доступом в приміщенні, що охороняється. В цьому випадку Аліса - це пропуск, виконаний у вигляді інтелектуальної картки, а Боб - комп'ютер охорони. У такій ситуації головне завдання - забезпечити коректність схеми аутентифікації, а захищатися від нечесного перевіряючого безглуздо. Що ж стосується якості нульового розголошення щодо чесного перевіряючого, то воно видається далеко не зайвим, так як дозволяє забезпечитися від противника, який може спробувати підслуховувати сеанси виконання протоколу з метою виготовлення

фальшивого пропуску.

Перш ніж завершити розмову про протоколи аутентифікації, необхідно підкреслити, що ніяке математично строго доведена властивість криптографічного протоколу не може гарантувати його безпеку у всіх випадках життя. Справді, навіть протокол доказу з нульовим розголошенням не захищає від наступної атаки на схему аутентифікації, відомої в криптографічній літературі під назвою «мафіозна загроза». У даному сценарії є чотири учасника: A, B, C і D. Припустимо, що A зайшов в кафе випити чашечку кави і розплачується за допомогою кредитної картки. Для виконання будь-якої банківської операції картка повинна ідентифікувати себе, тобто виконати протокол аутентифікації. Але власник кафе, B, - мафіозі, спільник якого C у той же самий момент знаходиться в магазині ювеліра D і намагається купити діамант, також за допомогою кредитної картки. При цьому C «представляється» як A, а D просить довести це за допомогою протоколу аутентифікації. Пристрій зчитування для карток у B і картка у C - це спеціально виготовлені приймально-передавальні пристрої, які лише пересилають повідомлення між A і D. У результаті A, обмінюючись повідомленнями з B, насправді ідентифікує себе для D. Отже, тільки математичного обґрунтування стійкості того чи іншого криптографічного протоколу недостатньо. Для практичного застосування конкретного протоколу потрібні ще зусилля фахівців в області інформаційної безпеки з аналізу умов його застосування.

7.5 Захищені обчислення

До захищених обчислень відносяться різні протоколи, метою яких є спільне обчислення деякого значення таким чином, щоб приховати це значення від всіх або деяких учасників. Розглянемо два протоколи захищених обчислень.

7.5.1 Приховування інформації від оракула

Нехай учасник B - оракул, який вміє обчислювати значення деякої важко обчислюваної функції f. A хоче звернутися до B за обчисленням $f(x)$, але не хоче розкривати значення x.

Параметри протоколу: конструкція (K, E, D), що відповідає вимогам до криптосистем з секретним ключем і з функцією дешифрування, що задовольняє умові:

$D_k(f(E_k(x))) = f(x)$ (*)

Алгоритм протоколу:

- A вибирає випадковим чином ключ k, шифрує x на цьому ключі: $E_k(x)$, і посилає його B.
- B обчислює значення $f(E_k(x))$ функції від зашифрованого значення і відсилає його A.
- A дешифрує отримане значення і отримує за умовою (*) значення функції.

Криптосистема з відкритим ключем тут не потрібна, так як не потрібно пересилати оракулу ні значення ключа, ні алгоритм дешифрування. Основна умова - наявність функцій шифрування і дешифрування, що задовольняють умові (*).

Прикладом такого протоколу може бути протокол обчислення важко обчислюваної функції $f(x)$ - дискретного логарифма x в Z^*_p по основі g, (g – який утворює елемент Z^*_p). У цьому випадку функції шифрування і дешифрування можна вибрати:

$E_k(x) = x\, g^k \bmod p$ і $D_k(y) = (y\text{-}k) \bmod (p\text{-}1)$, $k \in Z^*$

7.5.2 Завдання про двох мільйонерів

Два мільйонера хочуть з'ясувати у кого більше мільйонів, але так, щоб ніхто не дізнався, скільки їх у іншого.

Параметри протоколу:

Нехай a, b - секретні значення учасників А і В.

У протоколі потрібна криптосистема (K, E, D) з відкритим ключем і число m, таке, що $a \le m$ і $b \le m$. Нехай k- ключ учасника В.

Алгоритм протоколу:

- А вибирає велике просте число x, обчислює $t = E_k(x) - a$ і посилає його В.
- В вибирає просте число p, яке задовольняє вказаним нижче умовам і обчислює послідовність чисел $S = (u_1, ..., u_m, p)$ наступним чином:

$$u_i = \begin{cases} z_i, i \le b \\ 1, i > b \end{cases}, \text{ де } z_i = D_k(t+i) \bmod p,$$

а p вибирається так, щоб для $\forall i \ne j$ виконувалося $|z_i - z_j| > 1$.

Зауважимо, що при цьому $y_a = x$.

- А посилає В повідомлення «$a \le b$», якщо $u_a = x \bmod p$ і «$a > b$» в іншому випадку.

Цей протокол має очевидні недоліки:

- Чим більше m, тим складніше підібрати p, що задовольняє умовам.
- Учасник А дізнається результат раніше, ніж В і може відмовитися від виконання 3-го кроку.

7.6 Спеціальні види електронного підпису

Для вирішення завдань аутентифікації і забезпечення цілісності інформації використовується цифровий підпис. Звичайні протоколи електронного підпису, що розробляються для практичного застосування, є неінтерактивними (тобто весь обмін повідомленнями полягає в передачі відправником одержувачу підписаного повідомлення). Це викликано насамперед вимогами до ефективності.

Однак виникають ситуації в яких до електронного підпису пред'являються додаткові вимоги. Розглянемо спеціальні види електронного підпису, в яких використовуються багатораундові протоколи.

7.6.1 Сліпий підпис

Сенс протоколу сліпого підпису полягає в тому, щоб учасник В підписав запропоноване учасником А повідомлення, не отримавши інформації про це повідомлення. У «паперовому» документообігу така схема може бути реалізована за допомогою запечатаного конверту, в якому знаходиться документ, а поверх нього лист копірки. Такий підпис використовується, наприклад, у фінансовій криптографії для забезпечення властивості не відстежуваності електронних грошей.

У цифровому варіанті сліпий підпис являє собою окремий випадок приховання інформації від оракула. Повідомлення m шляхом шифрування упаковується в цифровий конверт $P(m)$, який підписується $S(P(m))$. При розкритті конверта виходить підпис для m: $P^{-1}(S(P(m)))$.

Наведемо реалізацію сліпого підпису RSA.

Протокол створення сліпого підпису RSA під повідомленням $m \in Z_m$

Ключі учасника B: (n, e) - відкриті і (n, d) - секретні.

- A → B $P(m) = mr^e \bmod n$, $r \in Z^*$

- B → A $S(P(m)) = P(m)^d \bmod n$

A обчислює $P^{-1}(S(P(m))) = r^{-1} S(P(m)) = m^d \bmod n$

Як правило, необхідно, щоб B підписував тільки повідомлення з деякої множини «прийнятних» повідомлень M, але не знав, яке саме повідомлення він підписав.

Протокол створення сліпого підпису RSA під прийнятним повідомленням $m \in M$.

Ключі учасника B: (n, e) - відкриті і (n, d) - секретні.

Нехай $m_1, \ldots, m_k \in M$ - повідомлення, для одного з яких буде отримано цифровий підпис.

1. A → B $(P_1(m_1), \ldots, P_k(m_k))$, де $P_i(m_i)=m_i^e \bmod n$, $r_i \in Z^* \; \forall i=1,\ldots,k$

- B → A вибирає номер конверта $j \in \{1, \ldots, k\}$, який підпише.

- A → B $\forall (r_i)\ i \neq j$, A, A відкриває всі конверти, крім j-го, B перевіряє, що $\forall i \neq j$ повідомлення прийнятні $m_i \in M$.

- B → A $S P_j(m_j)) = P(m_j)^d \bmod n$.

A розкриває підписаний конверт, тобто обчислює підпис для m:
$P^{-1}(S(P(m_j))) = r^{-1} S(P(m_j)) = m_j^d \bmod n$
Імовірність того, що A вдасться отримати підпис під неприйнятним повідомленням, не перевищує k^{-1}.

7.6.2 *Невидимий підпис*

Можна зробити так, щоб, той хто доводить підпис учасника P (prover) не могла бути перевірена перевіряючим V (verifier) без участі P. При цьому вийде так званий невидимий підпис або незаперечний підпис (undeniable signature, термін ввів David Chaum, вперше запропонував таку схему).

Застосування невидимою підписи:

2 надання фірмою можливості перевірити справжність розповсюджуваного програмного продукту тільки зареєстрованим користувачам;

3 здійснення без паперового документообігу всередині організації так, щоб в разі витоку документа за межі організації не можна було довести його справжність.

Параметри системи:
Група Z_p, де p - просте число.

Нехай $g \in Z_p$ – який утворює елемент (ділить p -1).

Ключі учасника P:

$x \in Z_p$ - закритий ключ,

$y = g^x$ - відкритий ключ.

Створення підпису:

$z = S(h) = h^x \bmod p$, де $h \in Z_p$ -хеш-функція повідомлення.

2 P → V число $w = h^a g^b \bmod p$, де $a,b \in Z_p$ - вибрані P числа.

3 V → P пару чисел (r,s), де $r = wg^t = h^a g^{b+t} \bmod p$.

де $t \in Z_p$ - обране V число і $s = r^x \bmod p$.

4 P → V пару чисел (a, b)

і P перевіряє, що $w = h^a g^b \bmod p$.

5 V → P число t

і V перевіряє, що $r = h^a g^{b+t}$ и $s = z^a y^{b+t} \bmod p$.

Твердження (*надійність протоколу*). Навіть володіючи необмеженою обчислювальною потужністю, P не може з ймовірністю, більшою p, дати вірні відповіді при невірному підпису.

Твердження (*безпека протоколу*). В ході протоколу не відбувається розкриття інформації про закриті ключі P, оскільки V може самостійно побудувати всі повідомлення протоколу з тим же розподілом, але без участі P.

7.7 *Поділ секрету*

Криптографічні схеми поділу секрету були незалежно відкриті Шаміром (A. Shamir) і Блеклі (G. R. Blakley). Основне призначення протоколів або схем поділу секрету - управління ключами. У багатьох практичних додатках доступність інформації залежить від одного-єдиного секретного ключа. Якщо ключ будь-яким чином втрачено (втрачений носій із записаним ключем, зруйнована пам'ять, в якій зберігається ключ, і т.д.), доступ до інформації блокується. Схеми поділу секрету дозволяють вирішити сформульовану проблему. Ідея полягає в поділі секретного ключа на компоненти з подальшим їх розподілом серед легальної спільноти учасників. Відновлення ключа можливо тільки в тому випадку, якщо деяка легальна коаліція учасників об'єднає свої ключові компоненти.

З точки зору математики завдання поділу секрету є перш за все комбінаторним завданням. Відзначимо, що перша постановка задачі була сформульована у вигляді такої комбінаторної проблеми. Розглянемо ситуацію, в якій одинадцять вчених працюють над секретним проектом. Робочі документи проекту замкнені в сейфі. Умова завдання - тільки шість і більше вчених, об'єднавшись і пред'явивши ключі, можуть відімкнути сейф і отримати доступ до документів.

Постановка завдання зводиться до оцінки найменшого числа замків сейфа і найменшого числа ключів у кожного з учених. Як рішення пропонується наступне міркування. За умовою існує як мінімум один замок, який не може бути відкритим жодним вченим з довільної п'ятірки. Таким чином, кожен з шести вчених, які залишилася, має ключ від цього замка. Причому немає необхідності в більш ніж одному подібному замку для коаліції з п'яти вчених. Таким чином, мінімальне число замків і ключів оцінюється як $C^5_{11} = 462$ і $C^5_{11-1} = 252$ відповідно. Очевидно, що число поєднань

пропоноване комбінаторне рішення є неприйнятним з практичної точки зору.

Криптографічні схеми поділу секрету пропонують інше рішення проблеми. Розглянемо її як *груповий криптографічний протокол* з деякою множиною учасників.

Зазвичай в протоколах ми розглядали пару взаємодіючих учасників (A, B). У групових протоколах замість A розглядатимемо групи учасників (A_{i1}, ..., A_{ik}, C), де учасники вибираються з множини можливих учасників U = {A_1, ..., A_l} (їх іноді називають процесорами). При цьому всі A_i і B взаємодіють тільки з виділеним учасником C. Учасник З називається *комбінатором* (combiner) або *дилером* (іноді, лідером) і доданий для досягнення *анонімності*, щоб B не міг дізнатися про склад групи A_i.

На множині U виділяється *структура доступу* Г - множина підмножин U, званих *дозволеними коаліціями*. Множина Г зазвичай монотонно по включенню: якщо $V \subset V'$ і $V' \in$Г, то $V \in$Г. Протокол повинен бути працездатний, якщо $\{A_{i1}, \ldots, A_{ik}\} \in$ Г.

Як і в інших типах криптографічних протоколів, в протоколі поділу секрету учасники, взагалі кажучи, не довіряють один одному, і кожен з них може виявитися противником, в тому числі і дилер. Груповий криптографічний протокол повинен відповідати наступним властивостям.

2 *Надійність*. Для будь-якої дозволеної коаліції $\{A_{i1}, \ldots, A_{ik}\} \in$ Г протокол $((A_{i1}, \ldots, A_{ik}, C), B)$ задовольняє вимогам надійності та безпеки.

3 *Групова безпека*. Для будь-якої невирішеної коаліції $\{A_{i1}, \ldots, A_{ik}\} \notin$ Г ніякі *змовники* $A_{i1}^*, \ldots, A_{ik}^*, C^*$ такі, що будь-який A_i^* і C^* має доступ до секретів всіх A_i і C, не можуть порушити надійність системи.

Надалі будемо розглядати коаліції учасників як підмножини множин номерів учасників I∈Г⊆{1, ...,*l*} = L.

Крім схем поділу секретів групові протоколи найчастіше використовуються для вирішення наступних завдань:

4 групове обчислення значення деякої функції - може застосовуватися для побудови групових криптосистем, групових підписів, групових ідентифікацій /протоколів конфіденційних обчислень/;

5 групова генерація псевдовипадкових чисел;

6 анонімна пошта і віддалене таємне голосування.

Але коло додатків схем поділу, який перевіряється, секрету значно ширше.

7.7.1 *Порогова схема поділу секрету*

Мета протоколу *поділу секрету* (*secret sharing*) - забезпечити відновлення секретного значення S∈К тільки дозволеною коаліцією учасників, кожен з яких має значення, званим *часткою* (*share*) або *тінню* (*shadow*) секрету $s_i \in$T.

Передбачається, що схема поділу секрету включає *n* учасників і керується авторизованим дилером. Основна функція дилера - поділ секрету на *n* компонентів («тіней») і розподіл їх серед учасників так, що будь-які *m* (і більше) учасники, зібравшись разом і пред'явивши тіні, можуть відновити секретний ключ. Причому будь-які (*m*-1) і менш учасників не можуть цього зробити.

Криптографічні схеми поділу секрету відомі також під назвою *m*-з-*n*-схем або (*m*, *n*) -порогових схем.

Визначення. Множини $\{s_1, \ldots, s_n\}$, які задовольняють наступним двом умовам, називаються (*m*, *n*) -пороговою схемою поділу секрету.

- Секрет S легко може бути обчислений за будь-яким m значенням s_i.

- Знання будь-яких (m-1) значень s_i не дозволяють визначити секрет S.

Компроміс між крипостійкістю і гнучкістю схеми регулюється вибором параметрів m і n.

Протокол складається з двох фаз.

На *фазі поділу секрету* дилер, знає деякий секрет S, генерує n частку секрету s_1, ..., s_n і посилає s_i процесору A_i по захищеному каналу зв'язку.

На *фазі відновлення секрету* будь-яка підмножина з не менш ніж m процесорів однозначно відновлює секрет, обмінюючись повідомленнями по захищеним каналам зв'язку. А будь-яка підмножина з не більш ніж (m-1) процесорів не може відновити секрет.

Визначення. Схемою поділу секрету з множиною секретів K і множиною частки T називається пара алгоритмів (D, R), таких, що:

- D (S) - імовірнісний поліноміальний алгоритм роздачі, який виходячи з секрету $s \in K$ обчислює набір часток учасників $s_i \in T$.

- R ($I, x_1, \ldots, x_{|I|}$) - детермінований поліноміальний алгоритм відновлення секрету такий, що:

$D(S) = (s_1, \ldots, s_n) \Rightarrow S = R(I, x_1, \ldots, x_{|I|})$

Для кожної дозволеної коаліції I алгоритм задає функцію:

$\phi_I(x_1, \ldots, x_{|I|}) = R(I, x_1, \ldots, x_{|I|})$

для кожної невирішеної коаліції I всі можливі значення S рівноімовірні: $\forall I \notin U$, $\forall x_i \in T$, $P(s=x \mid \forall i \in I, s_i = x_i) = P(s=x)$.

Визначення. Схема поділу секрету називається *досконалою*, якщо довільна коаліція або повністю розкриває секрет, або в результаті не отримує про нього ніякої апостеріорної інформації.

7.7.2 *Схема обчислення ключа доступу*

Схема обчислення ключа доступу при поділі секрету між двома учасниками.

Розглянемо найпростіший варіант досконалої схеми. Нехай є первинний секретний ключ і два учасника, що утворюють легальну коаліцію для отримання доступу до секретної інформації. Жоден з учасників не знає первинного секретного ключа, і тільки об'єднання тіней, які мають учасники, дозволяє відновити первинний ключ і виконати дешифрування.

Практична реалізація ідеї заснована на властивостях арифметики по модулю 2. Розглянемо первинний ключ S як послідовність двійкових символів (біт) довжини n.

Тоді тінь першого учасника s_1 обчислюється шляхом побітного підсумовування первинного ключа і шумової (випадкової) послідовності s_2 двійкових символів довжини n.

Та ж шумова послідовність s_2 використовується в якості тіні іншого учасника. Отже, первинний ключ S може бути обчислений підсумовуванням по модулю 2 тіней учасників:

$S = s_1 + s_2$

Оцінка трудомісткості розкриття первинного ключа для кожного з учасників, так само як і для нелегального зловмисника, становить $2n$ спроб дешифрування в гіршому випадку і $2n$ -1 в середньому.

7.7.3 Схема Шаміра

Схема поділу секрету, запропонована Шаміром, побудована за принципом поліноміальної інтерполяції. Нехай заданий багаточлен ступеня (m-1) над кінцевим полем Галуа GF (q) з q елементів (GF (q) = Z_q, де - q просте).

Секретний ключ задається вільним членом a_0, всі інші коефіцієнти багаточлена - випадкові елементи поля. Поле GF (q) відомо всім учасникам. Кожна з n тіней є точкою (x_i, y_i) кривої, що описується багаточленом $P(x)$, $x_i \neq 0$.

Скориставшись інтерполяційною формулою Лагранжа, наведеною нижче, можна відновити вихідний багаточлен (і, отже, секрет a_0,) по будь-яким m точках (тіням).

При цьому ймовірність розкриття секрету в разі довільних (m -1) тіней оцінюється як q-1, тобто в результаті інтерполяції по (m -1) точці секретом може бути будь-який елемент поля з однаковою ймовірністю. Отже, схема Шаміра є досконалою.

На Рис. 7.1 представлений спеціальний випадок; m = 2 (для відновлення секрету необхідно дві тіні).

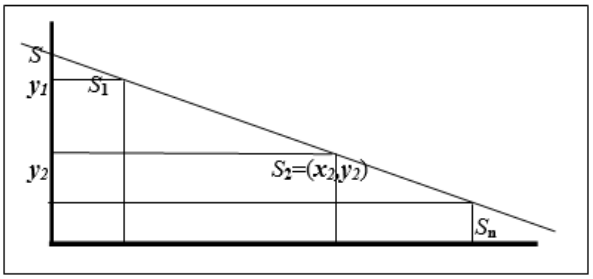


Рис. 7.1. Відновлення секрету по двом тіням

Таким чином, двочлен описує деяку пряму, яка перетинає з віссю у в точці s (секрет). Кожна тінь - точка на прямій. Отже, секрет може бути відновлений за двома довільними тінями, так як для однозначного визначення прямої достатньо двох довільних точок.

У разі завдання однієї тіні в якості шуканого секрету може бути обрана будь-яка точка на осі y, так як через одну точку можна провести безліч різних прямих, що перетинаються з віссю y в довільних точках.

7.7.4 Схема Блеклі

Схема поділу секрету Блеклі має геометричну природу. Секрет є точкою в m-вимірному просторі. Відзначимо, що для випадку m > 2 всі геометричні побудови виконуються над кінцевим полем GF (q). Кожна з n тіней задається як гіперплощина в m-вимірному просторі. Визначення секрету зводиться до знаходження точки перетину m гіперплощин.

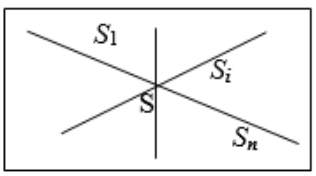


Рис. 7.2 Схема Блеклі

На Рис. 7.2 представлений спеціальний випадок схеми Блеклі. Кожна тінь - пряма, що проходить через цю точку. Для відновлення секрету S (точки на площині) необхідно мати принаймні дві тіні.

7.7.5 Метод «розшаровування» зображення

Наор і Шамір розробили оригінальний варіант схеми поділу секрету. Автори схеми запропонували наступну постановку задачі. Нехай є секретне зображення (в деякому

графічному форматі), яке необхідно розподілити серед n учасників.

Для цього зображення «розшаровується» на n складових (тіней) таким чином, що об'єднання будь-яких m з них дозволяє відновити зображення. Зрозуміло, що ні одна з тіней не дає уявлення про вихідне зображення. Більш того, ніякі (m-1) і менш тіней не дозволяють відновити вихідне зображення. Можливий варіант конструктивної реалізації схеми полягає в «розшаровуванні» зображення на чорно-білі клітинки пікселі з подальшою їх обробкою.

Схема є досконалою і проста в реалізації. Додаткова модифікація дозволяє отримувати як тіні не «шумові», а цілком змістовні зображення (наприклад, зображення пейзажу або будівлі і т.п.), що дозволяє приховати сам факт поділу секрету.

7.7.6 Протокол, який перевіряється, поділу секрету

У звичайних схемах поділу секрету розглядається пасивний супротивник, а саме, противником є не більше ніж (m-1) процесорів, які, об'єднавши свої частки, намагаються отримати будь-яку інформацію про значення секрету.

Якщо ж на фазі відновлення секрету діє *активний противник*, можливі наступні ситуації:

- дилер намагається роздати процесорам неправильні тіні,
- процесори A_i мають на меті зірвати відновлення секрету S, посилаючи чесним учасникам замість частки секрету будь-якої іншої інформації.

Подібна дія призведе до того, що секрет ще не буде правильно відновлений. При цьому неможливо визначити, хто з учасників коаліції винен в цьому. У гіршому випадку учасники просто не виявлять, що відновлений секрет не відповідає вихідному розділеному секрету.

Для захисту в подібній ситуації використовується *протокол, який перевіряється, поділу секрету* або верифікована схема поділу секрету.

Можливе рішення полягає в застосуванні *доказу з нульовим розкриттям*, що дозволяє одному учаснику довести коректність виконаних ним обчислень і знання тіні, отриманої в результаті поділу секрету, іншому учаснику схеми, не розкриваючи при цьому самої тіні. Даний протокол дозволяє вирішувати наступну практичну задачу: кожен з учасників може переконатися в тому, що отримана ним тінь, об'єднана з тінями інших учасників, дозволяє гарантовано відновити розділений секрет.

7.7.7 Протокол, який перевіряється, поділу секрету за схемою Шаміра

Для прикладу розглянемо схему Шаміра і ситуацію, в якій коаліція з m учасників намагається відновити секрет S за умови, що один з учасників - брехун і надає «помилкову» тінь (не ту, що була отримана в результаті поділу секрету). Фаза поділу секрету починається з того, що дилер публікує секрет S в «зашифрованому» вигляді (точніше було б сказати - виконує прив'язку до рядка S, за аналогією з прив'язкою до біту).

Дилер вибирає випадковий поліном ступеня m-1

$Q(x) = a_0 + a_1 x + \ldots + a_{m-1}x^{m-1}$, де $a_0 = S$

обчислює $r_i = g^{ai} \bmod p$ ($i = 1,\ldots, n$) і публікує $r_1,\ldots, r_n$.

За допомогою цієї інформації кожен процесор A_j може перевірити, що значення s_j, отримане ним від дилера, дійсно є часткою секрету S. Для цього він повинен перевірити рівність

$g^{sj} = r_0 (r_1)^j \ldots (r_{m-1})^{jm-1} \bmod q$

Справді:

$$rr^{j}r^{j}\ldots r^{j} = g^{a0}g^{a1j}g^{a2j}\ldots g^{anj} = g^{a0+a1j+a2j+\ldots+am-1j} = g^{Q(j)} \bmod q$$

$$0\ 1\ 2 \qquad m-1$$

На *фазі роздачі секрету* дилер обчислює:

$s_j = Q(j), j = 1,...,n$

і посилає це значення (частки або тіні секрету) процесору A_j по захищеному каналу.

Конструкцію протоколу для фази відновлення секрету розглянемо в найбільш простому випадку, коли дилер чесний. На цій фазі кожен процесор A_j надсилає кожному другому процесора A_j свою частку s_j.

Всякий чесний учасник A_i, отримавши деяке значення s_j від A_j, перевіряє це значення, як описано вище, і відкидає всі частки s_j, які не пройшли перевірку. Оскільки чесних учасників не менше m, A_i отримає принаймні m правильних частки секрету.

Використовуючи алгоритм відновлення секрету зі схеми Шаміра, відновимо значення S. Частки секрету є гіперплощинами в просторі багаточленів над полем Z_q. Для відновлення секрету будується єдина точка перетину m гіперплощин за допомогою інтерполяційного полінома Лагранжа:

$$Q'(x) = \sum_{i \in I} Q(x_i)\pi_i \text{, где } \pi_i = \prod_{j \in I, j \neq i} \frac{x - x_j}{x_i - x_j}$$

На відміну від звичайних схем поділу секрету стійкість даного протоколу ґрунтується на припущенні про обчислювальну складність завдання дискретного логарифмування. Тому, якщо в звичайних схемах поділу секрету потрібно, щоб будь-яка підмножина учасників, що не становить кворуму, не отримувало жодної інформації про секрет, то в багатьох схемах в яких перевіряється поділ секрету, така підмножина лише «не може відновити» секрет в тому сенсі, що для його відновлення потрібно вирішити деяку гіпотетично важку обчислювальну задачу. У розглянутому вище прикладі всякий учасник міг би дізнатися секрет S, якби він умів обчислювати дискретні логарифми.

7.7.8 Протоколи конфіденційного обчислення

Припустимо, що за допомогою протоколу поділу секрету розділені два секрета S_1 і S_2 і що обидва ці секрети є числами. Тепер уявімо собі ситуацію, що після цього було потрібно розділити секрет $S = S_1 + S_2$. Це може зробити дилер за допомогою того ж протоколу. А чи можуть процесори виконати те ж саме без участі дилера?

Нехай $Q_1(x) = a_0 + a_1 x + \ldots + a_{m-1}x^{m-1}$

і $Q_2(x) = b_0 + b_1 x + \ldots + b_{m-1}x^{m-1}$

- поліноми, які використовувалися для поділу секретів S_1 і S_2 відповідно.

Нехай $r_i^{(1)} = g^{ai} \bmod q$ і $r_i^{(2)} = g^{bi} \bmod q$, $i = 0,...,m\text{-}1$. - значення, що використовуються для перевірки правильності часток секрету.

І нехай $s_j^{(1)} = Q_1(j)$ і $s_j^{(2)} = Q_2(j)$, $j = 1,...,n$ - частки секретів S_1 і S_2, отримані процесором A_j.

Ясно, що $Q(x) = Q_1(x) + Q_2(x)$ -поліном ступеня m-1 і $Q(0) = S$.

Тому кожен процесор A_j може обчислити частку s_j секрету S просто по формулі:

$$s_j = s_j^{(1)} + s_j^{(2)}$$

Ці частки перевіряються за допомогою значень $r_i = r_i^{(1)} r_i^{(2)} \bmod q$.

Виконуючи такого роду обчислення над частками секретів, процесори можуть обчислити будь-яку функцію над кінцевим полем «перевіряємим чином». Типова задача тут така. Потрібно обчислити значення функції f на деякому наборі значень аргументів y_1, ..., y_k. За допомогою схеми, який $r_i^{(1)}$ перевіряється, поділ секрету обчислюється частки,

,..., $x^{(k)}$, (i = 1, ... n) цих значень. На початку виконання протоколу частка x_i відома процесору A_i і тільки йому. Протокол повинен забезпечувати обчислення значення:

$$f(x_1^{(1)},...,x_n^{(k)}) = f(y_1, \ldots, y_k)$$

таким чином, щоб для деякого параметра m:

1 в результаті виконання протоколу будь-яка підмножина з не більш ніж m-1 процесорів не отримувало жодної інформації про значення x_i інших процесорів (крім тієї, яка випливає з відомих їм часток) і значення функції $f(x_i^{(1)},..., x_i^{(k)})$

2 при будь-яких діях нечесних учасників інші учасники обчислюють правильне значення $f(y_1, ..., y_k)$, якщо тільки кількість чесних учасників не менше m.

Лекція №8 УПРАВЛІННЯ КЛЮЧАМИ

Під *ключовою інформацією* розуміється сукупність всіх діючих в ІС ключів. Якщо не забезпечено досить надійне управління ключовою інформацією, то заволодівши нею, зловмисник отримує необмежений доступ до всієї інформації.

Управління ключами являє собою найважчу частину криптографії. Проектувати криптографічні алгоритми та протоколи не просто, проте можна скористатися результатами теоретичних досліджень.

Криптоаналітики часто атакують симетричні і асиметричні криптосистеми, використовуючи недоліки схеми розподілу ключів. Іноді вкрасти ключ коштує набагато дешевше, ніж побудувати суперкомп'ютер або найняти криптоаналітика. Відомо, що деякі комерційні продукти не забезпечують надійного зберігання ключів.

Крім того, якщо для забезпечення конфіденційного обміну інформацією між двома користувачами процес обміну ключами тривіальний, то в ІС, де кількість користувачів становить десятки і сотні, управління ключами - серйозна проблема.

Управління ключами - інформаційний процес, що включає в себе три елементи:

- генерацію ключів;
- накопичення ключів;
- розподіл ключів.

Розглянемо, як вони повинні бути реалізовані для того, щоб забезпечити безпеку ключової інформації в ІС.

8.1 Генерація ключів

Безпека криптосистеми зосереджена в ключі. Якщо використовується криптографічно вразливий процес генерації ключів, то криптосистема в цілому вразлива. Криптоаналітику не потрібно займатися криптоаналізом алгоритму криптографічного перетворення, досить проаналізувати алгоритм генерації ключів.

8.1.1 Розмірність ключового простору

Один з основних параметрів стійкості криптосистеми - розмірність ключового простору. Він забезпечує захист від силової атаки, заснованої на повному переборі всіх можливих ключів. Наприклад, DES використовує 56-бітовий ключ. Якщо будь-яка правильно задана послідовність з 56 біт може бути ключем, то існує 2^{56} (10^{16}) можливих

ключів. У таблиці 8.1 наведені оцінки розмірності ключового простору для найбільш часто зустрічаючих алфавітів.

У таблиці 8.2 наведено час, необхідний для здійснення силової атаки при продуктивності один мільйон ключів в секунду. При такій продуктивності можливо розкрити ключі, що складаються з цифр і символів нижнього регістру довжиною до 8 байтів, алфавітно-цифрові ключі до 7 байтів, ключі з печатних символів і ASCII-символів до 6 байтів і ключі з 8-бітових ASCII-символів довжиною до 5 байтів.

Простір ключів	4 байта	5 байтів	6 байтів	7 байтів	8 байтів
Рядкові букви (26)	$4.6\ 10^5$	$1.2\ 10^7$	$3.1\ 10^8$	$8.0\ 10^9$	$2.1\ 10^{11}$
Рядкові букви і цифри (36)	$1.7\ 10^6$	$6.0\ 10^7$	$2.2\ 10^9$	$7.8\ 10^{10}$	$2.8\ 10^{12}$
Алф. і цифр, символи (62)	$1.5\ 10^7$	$9.2\ 10^8$	$5.7\ 10^{10}$	$3.5\ 10^{12}$	$2.2\ 10^{14}$
Печатки, символи (95)	$8.1\ 10^7$	$7.7\ 10^9$	$7.4\ 10^{11}$	$7.0\ 10^{13}$	$6.6\ 10^{15}$
ASCII (128)	$2.7\ 10^8$	$3.4\ 10^{10}$	$4.4\ 10^{12}$	$5.6\ 10^{14}$	$7.2\ 10^{16}$
8- бітові ASCII (256)	$4.3\ 10^9$	$1.1\ 10^{12}$	$2.8\ 10^{14}$	$7.2\ 10^{16}$	$1.8\ 10^{19}$

Таблиця 8.1. Оцінки потужності ключового простору

Простір ключів	4 байта	5 байтів	6 байтів	7 байтів	8 байтів
Рядкові букви (26)	0.5 сек	12 сек	5 хв	2.2 годин	2.4 дня
Рядкові букви і цифри (36)	1.7 сек	1 хв	36 хв	22 годин	33 дня
Алф. і цифр, символи (62)	15сек	15 хв	16 годин	41 день	6.9 років
Печатки, символи (95)	1.4 хв	2.1 годин	8.5 дня	2.2 року	210 років
ASCII (128)	4.5 хв	9.5 годин	51 день	18 років	2300 років
8- бітові ASCII (256)	1.2 годин	13 днів	8.9 років	2 300 років	580000 років

Таблиця 8.2. Трудоємність силової атаки при виробн. 1 млн. ключів в секунду

8.1.2 Випадкові ключі

Не варто використовувати невипадкові ключі з метою легкості їх запам'ятовування. У серйозних ІС використовуються спеціальні апаратні і програмні методи генерації випадкових ключів.

Добрими ключами є послідовності випадкових бітів, створені деяким автоматичним процесом. Ідеальними генераторами є пристрої на основі "натуральних" випадкових процесів. Наприклад, генерація ключів на основі *білого радіошуму*. Іншим випадковим об'єктом - математичним - є десяткові знаки ірраціональних чисел, наприклад

π або *e*, які обчислюються за допомогою стандартних математичних методів.

Як правило, використовують датчики псевдовипадкових чисел (ПСЧ). Однак ступінь випадковості їх генерації повинна бути досить високою. В ІС з середніми вимогами захищеності цілком прийнятні програмні генератори ключів, які обчислюють ПСЧ як складну функцію від поточного часу і (або) числа, введеного користувачем. Всі можливі ключі при цьому повинні бути рівноімовірні.

Для генерації ключів важливо використовувати хороший генератор випадкових чисел, але набагато важливіше використовувати надійні процедури управління і перевірки ключів. Деякі алгоритми шифрування мають ключі, які мають ряд специфічних властивостей, які полегшують їх розкриття. Якщо ці ключі відомі, необхідно виконувати їх перевірку в процесі генерації. У разі виявлення слабкого ключа генерується новий ключ. У DES існує 16 «слабких» ключів на 2^{56} можливих, так що ймовірність отримання такого ключа досить мала.

Генерація ключів для асиметричних криптосистем складніше; часто ключі повинні володіти певними математичними властивостями (можливо, вони повинні бути простими числами, квадратичними відрахуваннями і т.п.). Стартові послідовності для генераторів ключів повинні бути випадковими.

8.2 Генерація ключів за стандартом ANSI X9.17

Стандарт ANSI X9.17 визначає спосіб генерації ключів (див. Рис.8.1). Спосіб підходить для генерації сеансових ключів або псевдовипадкових чисел. Для генерації ключів використовується DES-шифрування, але воно може бути легко замінено будь-яким іншим криптоалгоритмом.

Нехай $E_k(x)$ - це повідомлення x, зашифроване на спеціальному ключі k, передбаченому для генерації секретних ключів. V_0 - секретна 64-бітова стартова послідовність. T - мітка часу. Для генерації випадкового ключа R_i необхідно виконати наступні обчислення:

$$R_i = E_k(E_k(T_i) \oplus V_i)$$

Для генерації V_{i+1} необхідно обчислити:

$$V_{i+1} = E_k(E_k(T_i) \oplus R_i)$$

Для перетворення R_i в ключ DES необхідно видалити кожен восьмий біт. Довший ключ виходить шляхом конкатенації кількох коротких ключів. Так для отримання 128-бітового ключа необхідно спочатку згенерувати пару ключів, а потім виконати їх конкатенацію.

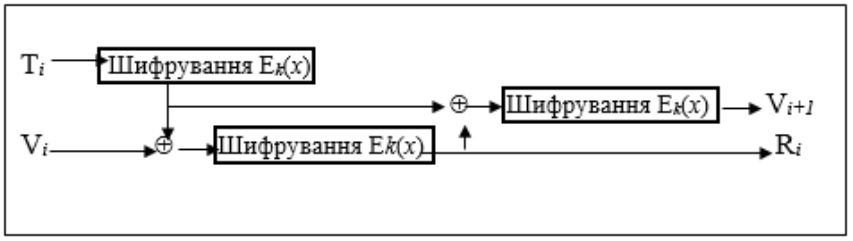


Рис. 8.1. Генерація ключів за стандартом ANSI X9.17

8.3 Нерівносильні ключі

Іноді необхідно (наприклад, якщо криптографічний пристрій потрапляє в руки до супротивника), щоб криптоалгоритм забезпечував адекватну криптостійкість тільки з секретними ключами деякого спеціального виду (легальні ключі), а з усіма іншими ключами (нелегальними) для шифрування використовувався б менш криптостійкий алгоритм. Можна зробити так, щоб ймовірність випадкової генерації легального ключа була дуже мала.

Простий спосіб полягає в генерації ключа, що складається з двох частин: безпосередньо ключа і деякої допоміжної фіксованої послідовності біт, зашифрованої на цьому ключі. Перед шифруванням блоку даних розшифровується допоміжна послідовність. Якщо результатом виявляється фіксована послідовність біт, то застосовується криптостійкий алгоритм шифрування, якщо немає, то менш криптостійкий алгоритм. Якщо алгоритм має 128-бітний ключ і 64-бітний розмір блоку, то довжина повного ключа 192 біта. Таким чином, існує 2^{128} різних легальних ключів, однак імовірність випадкового вибору легального ключа 2^{-64}.

8.4 Резервні ключі

Якщо ключ шифрування з якої-небудь причини буде втрачено, буде втрачена і вся зашифрована на ньому інформація. Дублювання ключів збільшує ризик їх компрометації.

Кращим рішенням є використання схеми поділу секрету. Коли Аліса генерує ключ, вона одночасно ділить його на кілька частин і потім посилає ці частини в зашифрованому вигляді різним уповноваженим особам. Жодна з цих частин сама по собі не є ключем, але, зібравши їх разом, можна відновити ключ. Згідно з ідеологією поділу секрету, ключ може бути відновлений навіть при втраті однієї або декількох частин.

Можна просто зберігати різні частини, зашифровані відкритими ключами відповідних довірених осіб, на своєму жорсткому диску. Таким чином, ніхто не бере участі в управлінні ключами, поки це не стане необхідним.

Інша схема резервування використовує інтелектуальні картки для тимчасового вручення ключів. Аліса може записати ключ, яким зашифрований її жорсткий диск, на інтелектуальну картку і передати її Бобу. Боб може використовувати картку для доступу до жорсткого диска Аліси, але так як ключ зберігається на картці, Боб не зможе його розкрити.

Крім того, такий спосіб допускає двосторонній контроль: Боб може упевнитися в тому, що ключ відкриває диск Аліси, а Аліса завжди може перевірити, чи був використаний ключ.

8.5 Завдання розподілу ключів

Як Боб, отримавши ключ, дізнається, що ключ переданий саме Алісою, а не будь-ким іншим, хто видає себе за Алісу?

- Якщо Аліса посилає свій ключ через довіреного кур'єра, то кур'єру повинен довіряти також і Боб.
- Якщо ключ зашифрований ключем шифрування ключів, то Боб повинен бути впевнений, що цей ключ є тільки у Аліси.
- Якщо використовується електронний цифровий підпис, Боб при перевірці підпису повинен довіряти відкритому ключу. Він також повинен бути впевнений, що парний секретний ключ Аліси не скомпрометований.
- Якщо Центр розподілу ключів (ЦРК) підписує відкритий ключ Аліси, Боб повинен бути впевнений в автентичності відкритого ключа ЦРК.

Асиметрична криптографія, що застосовується разом з електронними цифровими підписами і надійними ЦРК, сильно ускладнює підміну одного ключа іншим. Боб ніколи не може бути абсолютно впевнений в тому, що зловмисник його не контролює, однак Боб може знати напевно, що підміна ключа вимагатиме набагато більше ресурсів, ніж по силам дістати реальному зловмисникові.

Іноді ключі викривляються при передачі. Використання викривленого ключа при дешифруванні не дозволяє отримати відкритий текст, відповідний прийнятому шифртексту. Всі ключі повинні передаватися *з виявленням або виправленням помилок*. Одним з найбільш широко використовуваних методів є шифрування ключем деякої константи і передача перших декількох байтів отриманого шифртекста разом з ключем. Одержувач знає константу і виконує аналогічні дії - шифрує константу на отриманому ключі. Якщо прийнятий шифртекст збігається з результатом шифрування одержувача, то ключ був переданий без помилок. Імовірність помилки знаходиться в діапазоні від 2^{-16} до 2^{-32}.

8.6 Розподіл секретних ключів

Розподіл секретних ключів - найвідповідальніший процес в управлінні ключами. До нього висуваються дві вимоги:

1 Оперативність і точність розподілу.

2 Скритність ключів, які розподіляються.

Розподіл ключів між користувачами комп'ютерної мережі реалізується двома різними підходами:

- *Прямий обмін ключами* між користувачами інформаційної системи.
- Шляхом створення одного або декількох центрів розподілу ключів (ЦРК).

Ключі шифрування ключів, загальні для пари користувачів, зручно використовувати в мережах з невеликим числом абонентів, однак зі зростанням числа абонентів така схема швидко стає громіздкою. Так як кожна пара абонентів повинна обмінятися ключами, загальне число обмінів ключами в мережі з n абонентів дорівнює n (n-1)/2. У мережі з шістьма користувачами потрібно 15 обмінів ключами. У мережі з 1000 користувачів знадобиться вже близько 500 000 обмінів ключами. У цьому випадку розподіл ключів здійснюється за допомогою центрального сервера.

У разі прямого обміну ключами проблема полягає в тому, щоб надійно засвідчити справжність суб'єктів.

Відомий спосіб організації ЦРК на базі симетричних криптосистем має ряд недоліків:

По-перше, в центрі розподілу відомо, кому і які ключі призначені, і це дозволяє читати всі повідомлення, що циркулюють в ІС. Можливі зловживання значно впливають на захист.

По-друге, значний недолік полягає в тому, що для отримання секретного ключа абонента необхідно звернутися в ЦРК з відповідним запитом, тобто центр повинен обробляти потоки запитів в режимі *on-line*. Очевидно, що навантаження на ЦРК в мережі з великим числом абонентів може бути досить велика.

В обох підходах повинна бути гарантована справжність сеансу зв'язку. Це можна забезпечити двома способами:

1 *Механізм запиту-відповіді*, який полягає в наступному. Якщо користувач А бажає бути впевненим, що повідомлення, які він отримує від В, не є помилковими, він включає те що посилається для В повідомлення непередбачуваний елемент (запит). При відповіді користувач В повинен виконати деяку операцію над цим елементом (наприклад, додати 1). Це неможливо здійснити заздалегідь, так як невідомо, яке випадкове число прийде в запиті. Після отримання відповіді з результатами дій, користувач А може бути впевнений, що сеанс є справжнім. Недоліком цього методу є можливість встановлення закономірності (хоча і складної) між запитом і відповіддю.

2 *Механізм позначки часу* ("*тимчасової штемпель*"). Він має на увазі фіксацію часу для кожного повідомлення. У цьому випадку кожен користувач ІС може знати, наскільки "старим" є повідомлення, яке прийшло.

В обох випадках слід використовувати шифрування, щоб бути впевненим, що відповідь послана не зловмисником і штемпель позначки часу не змінений.

При використанні відміток часу постає проблема допустимого тимчасового інтервалу затримки для підтвердження автентичності сеансу. Адже повідомлення з "тимчасовим штемпелем" в принципі не може бути передано миттєво. Крім цього, комп'ютерний годинник одержувача і відправника не можуть бути абсолютно синхронізовані. Яке запізнювання "штемпеля" вважати підозрілим? У реальних ІС, наприклад в системах оплати кредитних карток, де використовується саме другий механізм встановлення автентичності та захисту від підробок, використовуваний інтервал становить від однієї до кількох хвилин. Велике число відомих способів крадіжки

електронних грошей засноване на "вклинюванні" в цей проміжок з підробленими запитами на зняття грошей.

8.7 Розподіл відкритих ключів

Для відкритих ключів проблема розподілу також актуальна. Основна відмінність відкритих ключів від секретних полягає в тому, що їх не треба ховати, але зате їх треба захищати від підробки. Без додаткових заходів безпеки обмін відкритими ключами вразливий з точки зору атаки, відомої під назвою «людина посередині». Зловмисник може підмінити відкриті ключі законних користувачів на свої власні і отримати повну можливість дешифрувати і підміняти всі повідомлення. Атака можлива навіть якщо відкриті ключі зберігаються в базі даних.

Сучасний підхід до вирішення проблеми заснований на властивостях асиметричних криптосистем (асиметрична схема управління ключами) і застосуванні спеціальних сертифікатів: для захисту від підміни відкриті ключі передаються разом з сертифікатами автентичності.

Сертифікат відкритого ключа абонента А створюється учасником протоколу X і має вигляд:

$C_{XA} = S_X$ ("A"|| час || k_A)

де S_X - цифровий підпис X, а k_A - відкритий ключ А.

Щоб можна було довіряти сертифікату C_{XA}, потрібні два, що не залежать один від одного, умови:

- впевненість в тому, що ключ, за допомогою якого створений цифровий підпис, дійсно належить учаснику X. Така впевненість може в свою чергу ґрунтуватися на іншому сертифікаті C_{YX}.
- довіра по відношенню до учасника X, тобто впевненість в тому, що X зі злого наміру або помилково не підпише підроблений ключ.

Є кілька схем використання сертифікатів.

8.7.1 Централізоване управління

Воно засноване на організації спеціальних довірених Центрів Сертифікації (ЦС), які видають сертифікати, що складаються з відкритого ключа абонента і унікальної ідентифікуючої інформації, скріплених цифровим підписом ЦС.

Абоненти обмінюються сертифікатами, отриманими від ЦС при встановленні захищеного режиму обміну, або поміщають їх в загальнодоступні мережеві довідники. Перед використанням відкритого ключа одержувача для шифрування адресованого йому повідомлення відправник може переконатися в його автентичності, перевіривши цифровий підпис сертифіката одержувача за допомогою відомого відкритого ключа центру сертифікації. Аналогічна перевірка відкритого ключа відправника виконується перед перевіркою цифрового підпису прийнятого одержувачем повідомлення.

- *Централізована схема* застосовується при організації захищеного WWW-обміну за допомогою протоколу SSL в продуктах Microsoft і Netscape. При цьому виділяється кілька спеціальних учасників, які називаються *Certification Authority*. Відкритий ключ будь-якого іншого учасника повинен бути підписаний одним з виділених учасників.
- *Ієрархічна схема* застосовується в системі захищеної електронної пошти PEM. Учасники утворюють дерево, причому кожен з них довіряє підписувати ключі всім своїм предкам в дереві і сам підписує ключі своїх безпосередніх нащадків.

8.7.2 Розподілене управління

У деяких випадках спосіб централізованого управління ключами працювати не буде. Можливо, не існує такого ЦС, якому довіряли б обидва абонента. Можливо, абоненти довіряють тільки своїм друзям або взагалі нікому не довіряють.

- *«Демократична» схема* (або розподілене управління ключами), застосовується в програмі PGP. Всі учасники рівноправні, і кожен може підписати ключ будь-якого учасника. Проблема автентичності відкритого ключа вирішується за допомогою поручителів. Поручителі - це абоненти криптомережі, які підписують відкриті ключі своїх друзів, партнерів і т.п.

Наприклад, коли Боб створює свій відкритий ключ, він передає копії ключа своїм друзям. Вони знають Боба, тому кожен з них підписує ключ Боба і видає Бобу копію свого підпису. Тепер, коли Боб пред'являє свій ключ чужій людині, наприклад Алісі, він пред'являє його разом з підписами цих поручителів. Якщо Аліса також знає цих поручителів і довіряє їм, у неї з'являється підстава для довіри Бобу. Для запобігання підміни ключа, перш ніж підписувати, поручитель повинен бути впевнений, що ключ належить саме Бобу.

Вигода цього механізму - відсутність ЦС, якому кожен повинен довіряти. А негативною стороною є відсутність гарантій того, що Аліса, яка отримала відкритий ключ Боба, знає когось із поручителів, і, отже, немає гарантій, що вона повірить в справжність ключа.

Пропонується довіряти ключу учасника А, якщо існує ланцюжок сертифікатів

$C_{X_0 X_1}, C_{X_1 X_2}, \ldots, C_{X_k A}$

При цьому слід бути впевненим в ключі X_0 і довіряти всім X_i підписувати ключі. Для перестраховки довжина ланцюжка не повинна перевищувати деяке невелике значення, наприклад 3.

Залишається єдина проблема - звідки береться перший відкритий ключ в ланцюжку. Цей ключ повинен бути переданий по захищеному каналу - при особистому контакті, спеціальною поштою, у вигляді підписаного документа з печаткою і т. п. Можна також передати ключ по електронній пошті, а по захищеному каналу - тільки *відбиток ключа*, результат обчислення криптографічно стійкою хеш функції від відкритого ключа.

Для отримання сертифіката в початковий момент часу потрібна організація секретного каналу для передачі в ЦС відкритого ключа та ідентифікуючої інформації.

При заміні відкритого ключа, необхідна для отримання сертифікату, інформація підписується на старому секретному ключі (якщо він не скомпрометований) і передається в ЦС по відкритому каналу. Результат перевірки підпису є критерієм для прийняття рішення про зміну відкритого ключа та видачі нового сертифікату.

Цікаве застосування можна знайти для сертифіката виду

$C_{AA} = S_A(\text{"A"} \,||\text{час}||\, k_A)$

У централізованій схемі він використовується для ключів, що належать виділеним учасникам, що досить безглуздо. У «демократичній» схемі такий сертифікат є сертифікатом відкликання ключа і публікується власником ключа, якщо закритий ключ було розголошено або викрадено. Ідея полягає в тому, що тільки власник ключа або викрадач можуть створити такий сертифікат.

Компрометація ЦС, так само як і в разі ЦРК, призводить до компрометації всієї мережі, однак пропонована асиметрична схема дозволяє зняти навантаження з ЦС, так як кількість запитів на отримання сертифікатів суттєво менше, ніж кількість запитів на встановлення захищеного режиму обміну інформацією на рівні абонентів.

8.7.3 Атаки на ЦС

Оскільки ЦС грає провідну роль в інфраструктурі криптомережі, розглянемо деякі прості атаки на ЦС на прикладі асиметричної схеми управління ключами.

I) Отримання сертифікату на чуже ім'я. Припустимо, Боб бажає видати себе за Алісу. Якщо Боб вміє підписувати повідомлення від особи Аліси, він може, наприклад, розпорядитися про зняття певної грошової суми з рахунку Аліси в банку. Для цього Боб генерує пару ключів (відкритий і секретний) і посилає відкритий ключ (по секретному каналу) в ЦС з твердженням, що він є Алісою. Боб досягне мети, якщо йому вдасться обдурити ЦС і отримати сертифікат на ім'я Аліси.

Для запобігання подібної атаки необхідно точно ідентифікувати особу запитувача. ЦС може, наприклад, вимагати особистої присутності і/або пред'явлення посвідчення особи. Кожен ЦС може мати свою процедуру аутентифікації, а також методи зберігання копій сертифікатів користувачів. Очевидно, що надійна організація служб ЦС визначає надійність криптомережі в цілому.

II) Розкриття секретного ключа центру сертифікації. Зловмисник, який розкрив секретний ключ ЦС, може підробляти сертифікати. Для запобігання витоку секретної інформації ЦС повинен зберігати свій секретний ключ, а також підписані на ньому сертифікати в спеціальному наднадійному, ударостійкому, захищеному від електромагнітних випромінювань, електронному приладі з енергонезалежною пам'яттю, так званий пристрій Підпису Сертифікатів (ППС).

Відомо, що силова атака на криптосистему RSA, введений ряд стандартів і широко використовується для шифрування і аутентифікації, зводиться до задачі розкладання великого двоскладного модуля на прості множники. Причому модуль відомий і входить у відкритий ключ ЦС. Дана обставина змушує ЦС використовувати наддовгі ключі (від 1000 біт і більше) і регулярно їх оновлювати.

Періодичність процедури поновлення ключів в ієрархічній структурі ЦС збільшується в напрямку від верхніх рівнів ієрархії до нижніх. Іншими словами, часта зміна ключів в ЦС верхнього рівня може бути непрактична, тому що веде за собою зміну ключів у великого числа користувачів криптомережі.

III) Підкуп співробітників ЦС. Інша модель атаки розглядає можливість підкупу Боба з боку Аліси. У цій ситуації Боб є співробітником ЦС, і мета підкупу полягає в отриманні Алісою сертифіката на ім'я Фреда. Отримавши сертифікат, Аліса зможе посилати повідомлення від імені Фреда, і всі ці повідомлення, в силу легальності виданого сертифіката, будуть адекватно сприйматися іншими користувачами криптомережі.

Таким чином, для досягнення мети необхідне об'єднання двох або більше зловмисників (коллаборативна атака). З огляду на можливість подібної атаки на ЦС, застосовують спеціальний метод розмежування доступу до ППС, заснований на відомій криптографічній техніці поділу секрету. Так, наприклад, для доступу до ППС може знадобитися участь декількох співробітників ЦС.

Процедура передбачає пред'явлення спеціальних секретних ключів (тіней основного ключа доступу), перевірку їх автентичності та дозвіл доступу в тому випадку, якщо число пред'явлених тіней не менш заданого порогового значення. Необхідно, однак, пам'ятати, що якщо потік запитів на видачу сертифікатів контролюється однією людиною, то описана вище атака може бути успішною.

IV) Підробка старих документів. Інша атака полягає в тому, що Аліса здійснює силову атаку для підбору модуля, що входить до складу відкритого ключа центру сертифікації. В результаті після закінчення певного часу (наприклад 15 років) їй вдається розкласти модуль і відновити старий секретний ключ ЦС. Термін дії цього секретного ключа вже закінчився, проте Аліса може підробити сертифікат 15-річної давності, який посвідчує помилковий відкритий ключ Боба. В результаті Аліса може підробити будь-який документ з підписом Боба 15-річної давності.

Зазвичай після закінчення певного терміну, наприклад двох років з моменту підпису, електронний документ стає недоступним. Однак існує ряд ситуацій, в яких електронні документи повинні надійно зберігатися тривалий час (довгострокові контракти і т.п.), при цьому повинна бути забезпечена можливість перевірки їх автентичності та цілісності.

Один із способів полягає у використанні крім звичайних ключів спеціального *наддовготривалого ключа*. Такі ключі мають значно більший модуль і зберігаються надійніше, ніж всі інші. Якщо дія наддовготривалого ключа закінчується через 50 років, то всі підписані на ньому документи повинні мати такий же термін дії.

При цьому виникає проблема, пов'язана з необмеженим ростом списку анульованих сертифікатів, оскільки наддовготривалі ключі, так само, як і всі інші, можуть бути скомпрометовані і повинні зберігатися протягом терміну їх дії.

Для усунення зазначеного недоліку пропонується реєструвати наддовготривалі ключі за допомогою стандартної процедури для звичайних ключів, наприклад з терміном дії протягом двох років. Після закінчення дворічного періоду наддовготривалі ключі, якщо вони не були скомпрометовані, повинні бути ресертифікованим на наступні два роки. Тепер скомпрометований наддовготривалий ключ повинен зберігатися в САС протягом двох років, а не п'ятдесят. При цьому, однак, зловмисник може добитися свого, втручаючись в роботу служби єдиного часу з метою зміни періодичності процедури ресертифікації.

Проблема може бути вирішена введенням спеціальної служби *тимчасових міток* (СВМ). Для цього всі довготривалі електронні документи повинні реєструватися із спеціальними і цифровими мітками, що видаються СВМ. Цифрові мітки дозволяють упевнитися в тому, що термін дії ключа (ключів), на якому було підписано довгостроковий документ, не закінчився. Крім того, цифрові мітки дозволяють встановлювати законність довготривалих електронних документів навіть у тому випадку, коли секретний ключ був скомпрометований після того, як документ був підписаний.

Для того, щоб проставити цифрову мітку на підписаному електронному документі, потрібно хешувати цей документ і надіслати отримане значення в СВМ. У відповідь отримуємо завірену підписом СВМ цифрову мітку, що складається з отриманого від неї значення хеш-функції, часу і дати, проставлених СВМ.

Хешування дозволяє не розкривати зміст документа (хеш-функцію неможливо звернути) в процесі взаємодії з СВМ. Очевидно, що передача інформації між користувачем і СВМ здійснюється з урахуванням вимог безпеки та може бути проконтрольована на цілісність і автентичність. Надалі для доказу дати і часу підпису документа Аліса надає сам документ і цифрову мітку.

Перевіряючий обчислює значення хеш-функції представленого документа, порівнює отримане значення з тим, яке зберігається в цифрі і мітці, і потім перевіряє запевнену цифрову мітку підпис СВМ. Доказ буде прийнято при збігу значень хеш-функції і валідності цифрового підпису СВМ.

Як узагальнення сказаного про розподіл ключів слід сказати наступне. Завдання управління ключами зводиться до пошуку такого протоколу розподілу ключів, який забезпечував би:

1. можливість відмови від центру розподілу ключів;
2. взаємне підтвердження автентичності учасників сеансу;
3. підтвердження достовірності сеансу механізмом запиту-відповіді;
4. використання при обміні ключами мінімального числа повідомлень.

8.8 Оновлення ключів

Розподіл ключів при частій їх заміні - складна практична задача. Особливо актуальна ця проблема при розподілі секретних ключів без використання ЦРК. Оригінальне рішення проблеми - використання "блукаючих ключів". Ідея методу досить проста. Після того, як ключ використаний в одному сеансі по деякому правилу, відомому і відправнику, і одержувачу, він змінюється іншим. Якщо правило зміни ключів обережно дотримується і відправником, і одержувачем, то в кожен момент часу вони мають однаковий ключ. Постійна зміна ключа ускладнює розкриття інформації зловмисником.

Основне завдання в реалізації цього методу - вибір ефективного правила зміни ключів. Найбільш простий шлях - генерація випадкового списку ключів. Зміна ключів здійснюється в порядку списку. Однак очевидно, що список доведеться якимось чином передавати. Необхідно, однак, пам'ятати, що секретність нового ключа визначається

секретністю старого. Якщо зловмисникові вдалося розкрити старий ключ, він зможе виконати оновлення ключів самостійно.

Інший варіант - використання математичних алгоритмів, заснованих на так званих перебираючих послідовностях. На множини ключів шляхом однієї і тієї ж операції над елементом виходить інший елемент. Іноді таку схему називають *схемою поновлення ключа*. Для цього необхідна хеш-функція, відома і Алісі, і Бобу. Якщо Аліса і Боб застосують до загального ключу одну і ту ж хеш-функцію, вони отримають однаковий результат. Потім вони можуть вибрати з результату потрібні їм біти і створити новий ключ. Найбільш доступним є використання полів Галуа. За рахунок зведення в ступінь породжуючого елемента можна послідовно переходити від одного числа до іншого. Ці числа приймаються в якості ключів. Ключовою інформацією в даному випадку є вихідний елемент, який перед початком зв'язку повинен бути відомий і відправнику, і одержувачу.

Надійність таких методів повинна бути забезпечена з урахуванням відомості зловмисникові використовуваного правила зміни ключів.

Цікавим і перспективним завданням є реалізація методу "блукаючих ключів" не для двох абонентів, а для досить великої мережі, коли повідомлення пересилаються між усіма учасниками.

8.9 Накопичення ключів

Під *накопиченням ключів* розуміється організація їх зберігання, обліку та видалення. Оскільки ключ є найпривабливішим для зловмисника об'єктом, який відкриває йому шлях до конфіденційної інформації, то питанням накопичення ключів слід приділяти особливу увагу.

Секретні ключі ніколи не повинні записуватися в явному вигляді на носії, який може бути зчитаний або скопійований. Кожна інформація про використовувані ключі повинна зберігатися в зашифрованому вигляді. Ключі, які зашифровують ключову інформацію, називаються *майстер-ключами*. Бажано, щоб майстер-ключі кожен користувач знав напам'ять і не зберігав їх взагалі на будь-яких матеріальних носіях, але це не завжди можливо.

Стандарт ANSI X9.17 визначає два типи ключів: *ключі шифрування ключів* і *ключі даних*. На ключах шифрування ключів шифруються ключі даних при передачі по відкритим каналам. На ключах даних шифруються самі повідомлення.

Дуже важливою умовою безпеки інформації є періодичне оновлення ключової інформації в ІС. При цьому перепризначатися повинні як звичайні ключі, так і майстер-ключі. Ключі шифрування ключів повинні розподілятися вручну і змінюватися досить рідко, так як розмір шифрованих повідомлень незначний. Однак необхідно пам'ятати, що при компрометації ключа шифрування ключів будуть скомпрометовані всі зашифровані на ньому ключі даних. Зміна джерел інформації відбувається набагато частіше. В особливо відповідальних ІС оновлення ключової інформації бажано робити щодня.

У досить складній ІС один користувач може працювати з великим об'ємом ключової інформації, і іноді навіть виникає необхідність організації баз даних ключової інформації. Такі бази даних відповідають за прийняття, зберігання, облік і видалення використовуваних ключів.

Іншим рішенням проблеми розподілу є розбиття ключа на кілька різних частин і передача їх по різним каналам. Якщо противник збере всі частини, крім однієї, він все одно не зможе розкрити ключ (за умови, що трудомісткість силової атаки при відновленні відсутньої частини ключа досить велика). Питання оновлення ключової інформації пов'язаний і з третім елементом управління ключами - розподілом ключів.

8.10 Зберігання ключів

Можна зберігати ключ в пам'яті користувача. Однак це не дуже надійно. Інше рішення - зберігання ключа у вигляді картки з магнітною смугою, пластикового ключа з вбудованою мікросхемою або інтелектуальної картки. Користувач може ввести свій ключ в систему, вставивши фізичний носій в пристрій, що зчитує, вбудований в автономний шифрувальний/дешифрувальний пристрій або підключений до комп'ютера. Практично будь-який здатний усвідомити, що таке фізичний ключ, яке його значення і як його захистити. Додання криптографічному ключу деякої фізичної форми робить зберігання і захист такого ключа інтуїтивно зрозумілим. Хоч користувач може використовувати ключ, він не знає його і не може його скомпрометувати. Він може використовувати його тільки тим способом і тільки для тих цілей, які визначені вектором контролю.

Розумний спосіб полягає в поділі ключа на дві половини, одна з яких зберігається в терміналі, а друга записана в пам'ять портативного носія. Втрата носія не призводить до компрометації криптографічного ключа. Компрометація також неможлива при несанкціонованому доступі до терміналу. Отже, компрометація носія або терміналу нічого не дає - для компрометації ключа необхідно дістати обидві його частини.

Ключі, які важко запам'ятати, можна зберігати зашифрованими. Наприклад, секретний ключ RSA може бути зашифрований ключем DES і записаний на жорсткий диск. Для відновлення RSA-ключа користувач повинен виконати DES-дешифрування. В ідеалі ключ ніколи не повинен надаватися поза шифрувального/дешифрувального пристрою у відкритому вигляді. Ця мета не завжди досяжна, але до цього потрібно прагнути.

8.11 Скомпрометовані ключі

Всі протоколи, методи і алгоритми сучасної криптографії мають сенс тільки в тому випадку, якщо ключ зберігається в секреті. Однак ключ Аліси може бути вкрадений, втрачений або скомпрометований іншим способом. Якщо скомпрометований ключ використовувався в *симетричній криптосистемі*, Алісі доведеться замінити ключ і сподіватися, що збиток мінімальний.

Якщо цей секретний ключ *асиметричній криптосистемі*, то все ускладнюється, так як парний відкритий ключ може зберігатися на багатьох серверах в мережі. Якщо зловмисник отримав доступ до секретного ключа Аліси, він може видати себе за неї і, як наслідок, читати адресовані їй повідомлення і підписуватися за неї.

Якщо ключ розподіляє ЦРК, то Аліса зобов'язана повідомити йому про факт компрометації. Якщо ЦРК не використовується, то їй слід сповістити всім абонентам які можуть отримувати від неї повідомлення. Відповідна служба криптомережі повинна оприлюднити той факт, що будь-яке повідомлення, отримане після компрометації ключа, є підозрілим і що ніхто не повинен посилати повідомлення Алісі, використовуючи відповідний відкритий ключ.

При використанні централізованої схеми розподілу відкритих ключів скомпрометовані ключі поміщаються в спеціальний *список анульованих сертифікатів* (CAC). Туди ж поміщаються і відкриті ключі з вичерпаним терміном дії. CAC служить механізмом, що дозволяє виключити можливість використання скомпрометованих ключів. CAC формується спеціальною службою ЦС і містить інформацію про сертифікати, виданих даними ЦС. Відзначимо, що список містить не самі скасовані сертифікати, а тільки інформацію, що дозволяє встановити, що сертифікат був скасований. Способи поширення CAC можуть бути різними: від безпосередньої розсилки користувачам криптомережі до організації в мережі спеціальних вузлів.

Таким чином, перед перевіркою підпису отриманого повідомлення і після верифікації сертифіката (перевірки підпису сертифіката на відкритому ключі ЦС) необхідно встановити, чи входить даний сертифікат в CAC. Якщо факт входження встановлено, прийняте повідомлення відкидається.

Добре, якщо Аліса знає, коли був скомпрометований її ключ. Механізм тимчасових

міток дозволяє абонентам визначити, які повідомлення законні, а якісь підозрілі. Додаткові ускладнення виникають, якщо Аліса не знає точно, коли її ключ був скомпрометований. Наприклад, Аліса може побажати відмовитися від контракту, так як він був підписаний замість неї людиною, яка вкрала у неї ключ. Якщо така відмова можлива, то хто завгодно зможе відмовитися від контракту, стверджуючи, що його ключ був скомпрометований до підписання контракту. Для вирішення подібних питань необхідно залучення незалежного арбітра. Цей приклад показує, як небезпечно мати один єдиний ключ (точніше, одну єдину пару ключів). Краще, щоб у Аліси були різні ключі (пари ключів) для різних додатків точно так само, як для різних замків потрібні різні фізичні ключі. Інші рішення проблеми включають біометричну аутентифікацію, обмеження на використання ключа, тимчасову затримку і додатковий підпис.

8.12 Час життя ключів

Жоден ключ шифрування не можна використовувати нескінченно. Час його дії повинно минати автоматично, подібно паспортам і ліцензіям. Причини цього в наступному.

Чим довше ви користуєтеся ключ, тим більше:

- ймовірність його компрометації. Якщо ключ використовується протягом року, то ймовірність його компрометації набагато вище, ніж якби його використовували тільки один день;

- втрати при компрометації ключа. Якщо ключ використовується для шифрування одного документа, то втрата ключа означає компрометацію тільки цього документа. Якщо той же самий ключ використовується для шифрування великих обсягів інформації, то його компрометація призводить до значних втрат;

- спокуса докласти необхідних зусиль для його розкриття. Розкриття ключа, використовуваного протягом доби, дозволить прочитати всі повідомлення, передані протягом доби. Розкриття ключа, використовуваного протягом року, дозволить зловмиснику отримати доступ до секретної інформації, переданої протягом року:

У ряді випадків трудомісткість криптоаналізу визначається кількістю шифртекста, отриманих в результаті шифрування на одному ключі.

Для будь-якого криптографічного додатка необхідна стратегія, що визначає допустимий час життя ключа. У різних ключів можуть бути різні часи життя.

Час життя ключів повинно визначатися в залежності від значимості і кількості даних, зашифрованих протягом заданого періоду:

- для криптографічного телефону має сенс використовувати ключ тільки протягом телефонної розмови, а для нової розмови використовувати новий ключ.

- чим більше швидкість передачі даних по каналу зв'язку, тим, можливо, частіше доведеться міняти ключ.

Необхідно *збалансувати ризики*, пов'язані із заміною ключа. Ключі *шифрування ключів* так часто змінювати не потрібно. При цьому шифр-текста для криптоаналітика утворюється небагато, а відповідний відкритий текст (випадкові ключі) неможливо вгадати. У деяких додатках ключі шифрування ключів замінюються лише раз на місяць або навіть раз на рік. Однак, якщо ключ шифрування ключів скомпрометований, потенційні втрати катастрофічні. Звичайно ж, втрата цього ключа означає втрату всіх ключів, які були на ньому зашифровані. Ключ шифрування ключів повинен зберігатися в безпечному місці.

При шифруванні *даних тривалого зберігання* ключі міняти часто не можна.

Щоденне дешифрування і повторне шифрування на новому ключі може привести до негативних наслідків - накопичення криптоаналітиком робочого матеріалу для подальшої атаки. Рішенням може послужити шифрування даних на деякому унікальному ключі з подальшим його шифруванням на ключі шифрування ключів.

Час життя секретних ключів *асиметричної криптографії* залежить від додатків. Секретні ключі для цифрових підписів можуть використовуватися роками (навіть протягом людського життя). У багатьох криптомережах термін дії секретних ключів обмежений двома роками. Старий ключ проте повинен зберігатися в секреті на випадок підтвердження підпису, зробленого під час дії старого ключа. Але для підписання нових документів повинен використовуватися новий ключ. Такий підхід дозволяє зменшити кількість матеріалу, яке криптоаналітик зможе використовувати для розкриття ключа.

8.13 Знищення ключів

Беручи до уваги, що ключі повинні регулярно змінюватися, старі ключі необхідно знищувати. З їх допомогою можна прочитати старі повідомлення, навіть якщо самі вони ніколи більше не використовуються.

Ключі повинні знищуватися надійно. Якщо ключ записаний в EEPROM-пам'ять, то для його знищення потрібно багаторазово перезаписати пам'ять. Якщо ключ записаний в пам'ять EPROM або PROM, то він повинен бути знищений разом з мікросхемою пам'яті.

Якщо ключ зберігається на жорсткому диску комп'ютера, дані відповідної ділянки накопичувача повинні бути багаторазово перезаписані. Серйозна проблема полягає в тому, що в комп'ютері ключі можуть бути розмножені і збережені в різних областях жорсткого диска. Будь-яка ОС, що реалізує будь-яку схему управління пам'яттю, постійно вивантажує програми з оперативної пам'яті на жорсткий диск і завантажує їх назад.

Для знищення ключів необхідно використовувати спеціальну програму, яка спочатку перевіряє наявність копій ключа на фізичному рівні, навіть в невикористовуваних блоках, а потім стирає їх.

8.14 Використання ключів

Виконання криптографічних програм під керуванням багатозадачних ОС пов'язано з певним ризиком. Неможливо сказати, коли операційна система зупинить працюючу програму, запише всі проміжні дані на жорсткий диск і передасть ресурси іншої задачі. Повертаючи управління перерваної програми і зчитуючи при цьому всю необхідну інформацію з жорсткого диска, операційна система не очищує пам'ять накопичувача.

Очевидно, що, якщо виконувалася програма шифрування, то крім самої програми на жорсткий диск будуть записані деякі дані, в тому числі і секретний ключ. Секретний ключ буде зберігатися на диску до тих пір, поки не буде виконано запис в цю ж область пам'яті.

У пріоритетному, багатозадачному середовищі для зниження ризику можна встановити високий пріоритет виконання завдання. Але навіть в цьому випадку надійність системи в цілому буде невисокою.

Апаратні реалізації надійніше. Багато з пристроїв шифрування розроблені так, що будь-яке втручання призводить до знищення ключа. Деякі пристрої, наприклад телефонні шифратори, використовують сеансові ключі. Сеансовим називається ключ, який використовується тільки для одного сеансу зв'язку - єдиної телефонної розмови - і потім знищується. Немає сенсу зберігати ключ після того, як він був використаний.

У деяких додатках необхідний контроль процесу використання сеансового ключа. *Сеансові ключі* можуть бути дозволені до використання тільки на певній машині або тільки в певний час.

Одна зі схем управління подібними обмеженнями передбачає додавання до ключа спеціального *вектора контролю*. Значення хеш-функції вектора контролю підсумовується по модулю два з головним ключем. Результат використовується як ключ для шифрування сеансового ключа. Отриманий шифртекст потім зберігається разом з вектором контролю. Для відновлення сеансового ключа потрібно обчислити значення хеш-функції вектора контролю і підсумувати результат з головним ключем. Отриманий ключ використовується для дешифрування і відновлення сеансового ключа.

Перевага цієї схеми полягає в тому, що довжина вектора контролю може бути довільною і що він завжди зберігається у відкритому вигляді разом із зашифрованим ключем. Такий спосіб не висуває вимог щодо захищеності апаратури і передбачає відсутність безпосереднього доступу користувачів до ключів.

8.15 Депонування ключів

У квітні 1993 року уряд США анонсувало новий метод криптографічного захисту інформації, що забезпечує високий рівень безпеки при передачі по відкритим каналам зв'язку, з одного боку, і відповідає вимогам національної безпеки, з іншого.

Метод заснований на застосуванні шифрувальної/дешифрувальної мікросхеми Clipper, розробленої за спеціальною технологією TEMPEST, що перешкоджає зчитування інформації за допомогою зовнішнього впливу і процедури депонування ключа, визначальної дисципліни розкриття унікального ключа мікросхеми.

Генерація та запис ключа виконуються до вбудовування мікросхеми в кінцевий пристрій. Не існує способу, що дозволяє безпосередньо прочитати ключ як під час, так і після закінчення технологічного процесу виробництва і програмування мікросхеми.

Ключ розбивається на два компоненти, кожен з яких шифрується і потім передається на зберігання довіреним агентам депозитної служби, які представляють собою урядові організації, що забезпечують надійне зберігання ключових компонентів протягом терміну їх дії. Агенти видають ключові компоненти тільки тоді, коли відповідний запит підтверджений вирішенню федерального суду. Отримані компоненти дозволяють відновити унікальний ключ і виконати дешифрування.

Технологія поділу ключа заснована на математиці поділу секрету. Нехай є первинний секретний ключ і два учасники, що утворюють легальну коаліцію для отримання доступу до секретної інформації. Останнє означає, що жоден з учасників не знає первинного секретного ключа і тільки об'єднання вторинних ключів, які мають учасники, дозволяє відновити первинний ключ і виконати дешифрування.

Практична реалізація ідеї. Нехай первинний ключ k - це послідовність бітів довжини n. Тоді вторинний ключ першого учасника

$$k_1 = k \oplus \xi$$

де ξ - шумова послідовність двійкових символів довжини n. Та ж шумова послідовність використовується в якості вторинного ключа іншого учасника $k_2 = \xi$. Отже, первинний ключ може бути вирахуваний підсумовуванням за модулем 2 вторинних ключів учасників:

$$k = k_1 \oplus k_2$$

Схема має абсолютну секретність - оцінка трудомісткості розкриття первинного ключа кожним з учасників становить $2n$ спроб дешифрування, в гіршому випадку $2n$-1 - в середньому. Дана схема є окремим випадком класичного методу поділу секрету.

Інший довгостроковий проект уряду США, що діє на підставі закону про комп'ютерну безпеку від 1987 р. пов'язаний з розробкою мікросхеми Capstone. Підтримка

проекту здійснюється національним інститутом стандартів і технології (NIST) і агентством національної безпеки (NSA).

У мікросхемі Capstone крім стандартних компонентів мікро-схеми Clipper реалізований алгоритм обміну відкритими ключами KEA (Key Exchange Algorithm), цифровий підпис DSA, хеш-функція SHA, алгоритм швидкого зведення в ступінь за модулем і генератор псевдовипадкових чисел на основі джерела чистого шуму. Всі криптоалгоритми мікросхеми Capstone мають 80-бітний секретний ключ. Основна область застосування мікросхеми Capstone - безпека електронних угод і платежів, а також ряд інших сфер в рамках національної інформаційної інфраструктури.

Спосіб застосування. Для захисту телефонних переговорів кожен з абонентів повинен мати спеціальний криптографічний пристрій, що містить Clipper, Capstone або будь-яку іншу аналогічну мікросхему. У пристрої має бути реалізований протокол, що дозволяє абонентам обмінюватися секретним сеансовим ключем, наприклад, за допомогою відомого методу цифрового конверта. Даний метод застосовується в пристрої захисту телефонних переговорів TSD (Telephone Security Device), розроблений компанією AT & T. Пристрій підключається щільно один до одного між телефонною трубкою і основним блоком і активізується натисканням кнопки.

БІБЛІОГРАФІЧНИЙ СПИСОК

5. Брюс Шнайдер. Криптографія: принципи і практика. – Київ: ВД "Професіонал", 2017. – 432 с.
6. Мороз, В. В. Основи криптографії та захисту інформації. – Київ: Кондор, 2012. – 200 с.
7. Шевченко, В. І. Криптографія та інформаційна безпека. – Харків: ХНУРЕ, 2016. – 300 с.
8. Камінський, О. О. Захист інформації: теорія та практика. – Київ: Либідь, 2015. – 400 с.
9. Лисенко, В. В. Методи і засоби криптографії. – Одеса: Одеська національна академія харчових технологій, 2018. – 250 с.
10. Романенко, А. В. Теоретичні основи криптографії. – Львів: Львівська політехніка, 2020. – 220 с.
11. Stallings, W. Cryptography and Network Security: Principles and Practice. – 7th ed. – Boston: Pearson, 2017. – 672 p.
12. Paar, C., & Pelzl, J. Understanding Cryptography: A Textbook for Students and Practitioners. – Berlin: Springer, 2010. – 362 p.
13. Stinson, D. R. Cryptography: Theory and Practice. – 3rd ed. – Boca Raton: Chapman & Hall/CRC, 2006. – 688 p.
14. NIST Special Publication 800-131A. Transitioning the Use of Cryptographic Algorithms and Key Lengths. – National Institute of Standards and Technology, 2019. – 31 p.
15. FIPS PUB 197. Announcing the Advanced Encryption Standard (AES). – National Institute of Standards and Technology, 2001. – 60 p.